FACHBEREICH ELEKTROTECHNIK UND INFORMATIONSTECHNIK

TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Diplomarbeit

Adaptive Analog-to-Digital Conversion and pre-correlation Interference Mitigation Techniques in a GNSS receiver

Thorsten Lotz

---

18. November 2008

Diplomarbeit

Adaptive Analog-to-Digital Conversion and pre-correlation Interference Mitigation Techniques in a GNSS receiver

Fachbereich Elektrotechnik und Informationstechnik
Lehrstuhl für hochfrequente
Signalübertragung und -verarbeitung
TECHNISCHE UNIVERSITÄT KAISERSLAUTERN

Thorsten Lotz

Tag der Ausgabe : 01. Juni 2008
Tag der Abgabe : 18. November 2008

Betreuer : Priv. Doz. Dr.-Ing. habil. Michael Meurer
Dipl.-Ing. Manuel Cuntz

Acknowledgements

I thank Dr.-Ing. habil. Michael Meurer for giving me the chance to make new experiences in the field of global navigation satellite systems and welcoming me at the Institute of Communications and Navigation of the German Aerospace Center (DLR) in Oberpfaffenhofen.

I also would like to thank my supervisor Dipl.-Ing. Manuel Cuntz for his constant support and guidance, for his advices and for the time he took for me.

I thank my family for their encouragement and support during the whole work.

Contents

List of Figures

List of Tables

1. Introduction

The discovery of navigation seems to have occurred early in human history. According to a Chinese storytelling, the compass was discovered and used in wars during foggy weather before recorded history. There have been many different navigation techniques to support ocean and air transportation. Satellite-based navigation started in the early 1970s. Three satellite systems were explored before the GPS program: the U.S. Navy Navigation Satellite System (also referred to as the Transit), the U.S. Navy's TIMATION (TIMe navigATION), and U.S. Air Force project 621B. The Transit project used a continuous wave (CW) signal. The closest approach of the satellite can be found by measuring the maximum rate of Doppler shift. The TIMATION program used an atomic clock that improves the prediction of satellite orbits and reduces the ground control update rate. The Air Force 621B project used the pseudorandom noise (PRN) signal to modulate the carrier frequency.

The GPS program was approved in December 1973. The first satellite was launched in 1978. In August 1993, GPS had 24 satellites in orbit and in December of the same year the initial operation capability was established. In February 1994, the Federal Aviation Agency (FAA) declared GPS ready for aviation use [Tsu00].

1.1. Satellite Navigation Systems

1.1.1. GPS

Nowadays there is only one fully operational worldwide global positioning system, the NAVSTAR *(NAVigation System with Time and Ranging)* Global Positioning System, generally referred to as GPS. It was formed out of the concept for the *Defense Navigation Satellite System* (DNSS) program of the U.S. Department of Defense.This concept based on the work of several agencies and was initialized in the early 1960's by the US military, the *National Aeronautics and Space Administration* (NASA) and the *Department of Transportation* (DOT). The GPS provides accurate, continuous, worldwide, three dimensional positioning and velocity information to

an unlimited amount of users. Its satellite constellation consists of 24 satellites, arranged in six orbital planes, with four satellites per plane. The status and health of each satellite is observed by a network of five monitor stations which are arranged along the equator around the globe. Each of the five monitor stations contain multiple GPS tracking receivers with redundant cesium standard clocks. In addition there are four ground antenna upload stations that have the capability of uploading navigation data to the satellites [BWP96].

The satellites, which are moving in space at a speed of about 4 km/s, are the references for the users's position determination. So the user position can be estimated with an error no larger than 15 meter for civil use based on prediction made 24-28 hours earlier and transmitted in the ephemeris data. *Time of arrival* (TOA) ranging is utilized to determine the distance between satellite and user. The signals transmitted by each satellite are referenced to onboard highly accurate atomic clocks which are synchronized to the GPS system time. Each transmitted satellite signal contains the transmission time according to the GPS system time and its own spread spectrum signature. This signature enables a precise estimation of position by measuring the transit time from a known satellite to the user. Further it enables all satellites to transmit at the same carrier frequency in the same frequency bands. Thus, a *Code Division Multiple Access* (CDMA) scheme is employed.

1.1.2. GLONASS

Since the GPS was developed during the cold war by the U.S., there was of course a similar system of their opponent, the UdSSR, called the GLONASS *(Global'naya Navigatsionnaya Sputnikovaya Sistema)*. Primarily, it was designed for military purpose, like the GPS. But since the dissolution of the UdSSR the GLONASS suffered from a lack of resources due to the changed political and economic climate, hence only a skeleton remained of the original system. But the Russian Federation undertakes new considerable efforts to achieve a fully operational system by 2011, today there are already 18 satellites back in orbit [Wik, GLONASS].

1.1.3. Galileo

In 1999 Europe started the development of an own *Global Navigation Satellite System* (GNSS), called Galileo. It will be a new GNSS with higher accuracy and multiple new services. Galileo will be established as a public-private partnership under civilian control. It is financed and managed by the *European Commission* (EC),

the *European Space Agency* (ESA) and the European industry. Galileo will be operational in 2013. It benefits from the technological advances in satellite navigation since the mid 1970's and will be compatible to GPS.

1.2. Problem of Radio Interference

Due to the fact that GPS was developed for the military purpose, it was especially designed to have high resistance against jamming. For this reason the GPS makes use of spread-spectrum techniques which allow the system to work even in presence of *radio frequency interference* (RFI) with low powers properly.

However, the performance of these spread-spectrum techniques to mitigate the impact of RFI on the transmitted information is limited. Especially strong CW interference can easily capture the receiver's phase-locked-loops and avoid a position estimation. Another issue arises due to the fact, that the correlator needs a digital signal to determine the navigation information. But in order to get that digital signal an *analog-to-digital-converter* (ADC) in combination with amplifiers, which fit the incoming signal into the ADC range, are necessary. Those amplifiers are normally set according to ratio between the received noise floor and the maximum quantization threshold of the ADC. The threat of RFI is that it has a greater power than the noise floor, so it can easily saturate and even damage the receivers electronics. Because of that RFI is a major source for degradation of the GPS accuracy and reliability.

For a description of possible RFI types see Chapter 4. The main objective of this work is to reduce their impact on the digital tracking loops and so to improve the *signal-to-noise ratio* (SNR) at the correlator output.

1.3. Objectives of the Study

Presently a GNSS receiver for safety of life applications is in development at the DLR institute of communications and navigation in Oberpfaffenhofen. In future, this receiver will be implemented in air traffic to support for example machine-aided landing. Thus, a high receiver robustness and accuracy even in presence of strong RFI is required, which makes it necessary to mitigate interference already before the signal is passed to the correlator.

Therefore, the objective of this diploma thesis is to develop and asses the performance of pre-correlation interference detection and mitigation techniques for the future implementation in the DLR GNSS receiver.

The developed algorithms will be directly implemented on an FPGA board, to which the amplified, bandlimited, sampled and quantized antenna input signal is driven. So there are no restrictions on the complexity of the developed algorithms, but there are some on the quality of the quantized signal, which are determined by the AGC and ADC. In the real world DLR GNSS receiver a 14 *bit* ADC and a 20 *dB* AGC are available for the A/D-conversion, on the basis of the AGC/ADC output signal digital interference detection and mitigation is performed. But note, to reduce the complexity of further signal processing units, the FPGA board output signal will only have a 6 *bit* resolution.

1.4. Organization of this thesis

Chapter 2 outlines the technical background of satellite navigation. Therefore, first the principles of satellite based position determination are discussed. Than, as GPS signals are used to verify the developed RFI mitigation techniques, the specifics of the GPS, especially its signal structure are given. In this context it is important to know about the auto-correlation properties of the C/A-code, as they retrieve the GPS signal from the noise. Afterwards the basic GNSS receiver structure is given and the fundamentals of acquisition and tracking are presented.

Chapter 3 examines the basics of multibit A/D-conversion and different gain controlling strategies which all have the same objective to reduce saturation effects in the ADC and to minimize quantization loss.

In Chapter 4, RFI is of major concern, therefore different types of interference and their sources are examined. The second part of this chapter reveals the effects of RFI on the GNSS receiver.

In order to deal with interference it needs to be detected first. Therefore Chapter 5 shows different interference detection strategies, which there are monitoring the AGC, the ADC bins distribution, power fluctuations in the time and power spectral density.

In Chapter 6 different pre-correlation interference mitigation techniques working in the time and frequency domain are presented. In this context temporal blanking and frequency excision techniques are of major concern.

Last but not least, in Chapter 7 the performance of the developed pre-correlation interference detection and mitigation techniques is assesed by using the DLR GNSS Software Receiver.

2. Technical Background

This section gives an insight into how satellite based navigation works. Therefore first of all it is discussed how to determine a position in a three dimensional space on the basis of the satellite constellation in view. Then the specifics of the American GPS NAVSTAR and its signal structure, with its power levels before and after the correlation process, are examined. At last that the basic GNSS receiver architecture is introduced and a short description of GPS signal acquisition and tracking is given.

2.1. Fundamentals of Satellite Navigation

2.1.1. Basic Concept for Determining the User Position

In order to determine a certain point in space the distances from this point to some known positions in space need to be measured. Therefore first consider the user position determination for a one-dimensional case. In Figure 2.1, the user position is on the x-axis. If the satellite position S_1 and the distance to the satellite x_1 are both known, the user position can be at two places, either to the left or right of S_1. In order to find the correct user position, the distance to another satellite with known position must be measured. In this figure, the position of S_2 and x_2 uniquely determine the user position U.

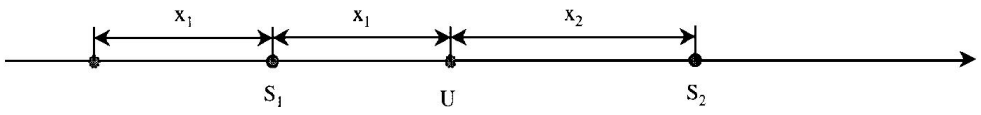

Figure 2.1.: One-dimensional user position determination [Tsu00].

The next Figure 2.2 shows the two-dimensional case. In order to determine the user position, three satellites and three distances are required. The trace of a point with constant distance to a fixed point is a circle in the two-dimensional case. Two

satellites and two distances give two possible solutions because two circles intersect at two points. A third circle is needed to uniquely determine the user position.

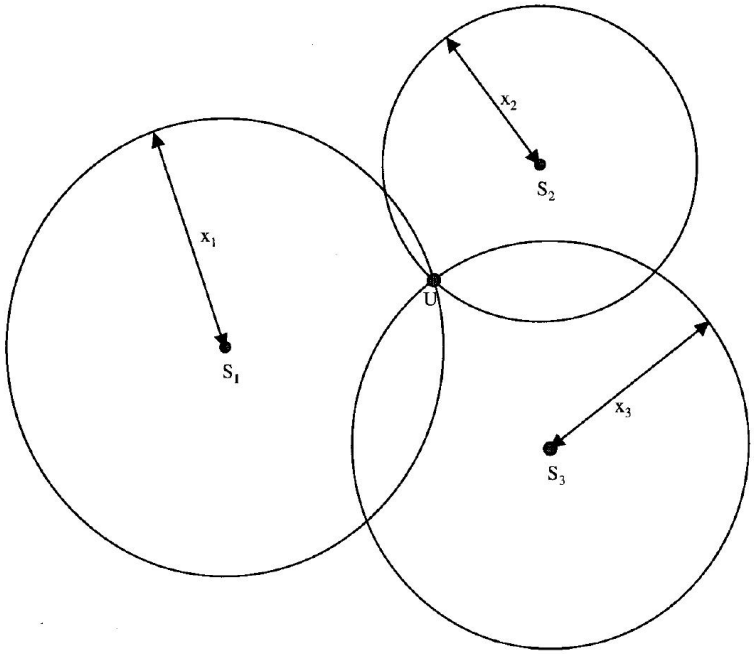

Figure 2.2.: Two-dimensional user position determination [Tsu00].

For similar reasons one might decide that in a three-dimensional case four satellites and four distances are needed. The equal-distance trace of a fixed point is a sphere in a three-dimensional case. Two spheres intersect to make a circle. This circle intersects another sphere to produce two points, see Figure 2.3. In order to determine which point is the user position, one more satellite is needed. However, in fact that the user is situated on earth's surface, the ambiguity of only using three satellites can be solved by taking the position closer to the surface.
In GPS the position of the satellite is known from the ephemeris data transmitted by the satellite. So one can measure the distance from the receiver to the known reference point which is the satellite position, therefore the user position can be determined.

In the above discussion, the distance measured from the user to the satellite is assumed to be very accurate and without error. However, the distance measurement between the receiver and the satellite is done by determining the time of how long a signal needs to travel from the satellite to the user. Therefore very accurate clocks are necessary. But in fact the receiver is not provided with a accurate clock, the user clock is usually offset from that of the satellite by a constant bias. In order to find the user position and with the knowledge that the user is on the earth surface, at least four satellites are needed in total to determine his position [Tsu00].

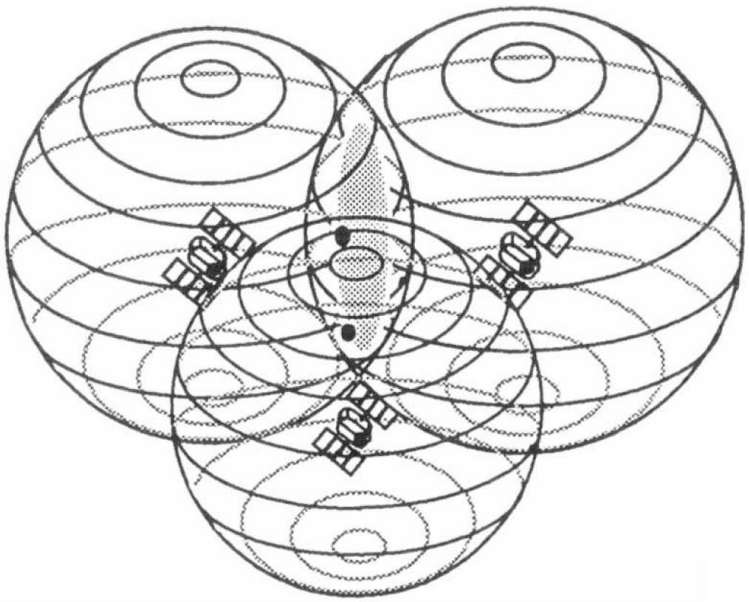

Figure 2.3.: User located at one of two points on shaded circle [Kap96].

2.1.2. Basic Equations for Finding the User Position

In this section the basic equations to determine the user position are presented. These are the basics only, therefore it is assumed, that the distances measured are accurate and under this conditions three satellites are sufficient. So in order to determine the user position one only has to measure the distances between known satellite positions to unknown user position. The three measured distances ρ_1, ρ_2 and ρ_3 fulfill the following equations

$$\begin{aligned}\rho_1 &= \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2},\\ \rho_2 &= \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2},\\ \rho_3 &= \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2}.\end{aligned} \tag{2.1}$$

Because there are three unknows and three equations, the values of x_u, y_u and z_u can be determined from these equations [Tsu00].

2.1.3. Measurement of Pseudorange

In GPS operation the satellite positions are given. This information is contained in the ephemeris data transmitted from the satellites. The distance from the user

to the satellites must be measured simultaneously at a certain time instance. Each satellite transmits a signal with a time reference associated with it. By measuring the time of the signal traveling from the satellite to the user the distance between them can be determined. So when the satellite sends its signal at a certain time t_s, the receiver will receive the signal at a later time t_u. The distance between the user and the satellite is

$$\rho_T = c(t_u - t_s), \tag{2.2}$$

where c is the speed of light, ρ_T is the true value of pseudorange from the user to the satellite, t_s is also referred as the true time of transmission, t_u is the true time of reception.

As already mentioned, in real world one cannot get the true time of transmission and arrival with only three satellites. So, the actual satellite clock time t'_s and the actual user clock time t'_u are related to the true time as

$$\begin{aligned} t'_s &= t_s + \Delta b, \\ t'_u &= t_u + b_{ut}, \end{aligned} \tag{2.3}$$

where Δb is the satellite clock error and b_{ut} is the user clock bias error. Beside the clock error, there are other factors affecting the pseudorange measurement. The measured pseudorange ρ can be written as

$$\rho = \rho_T + \Delta D - c(\Delta b - b_{ut}) + c(\Delta T + \Delta\ I + v + \Delta v), \tag{2.4}$$

where ΔD is the satellite position error effect on range, ΔT is the tropospheric delay error, ΔI is the ionospheric delay error, v is the receiver measurement noise error, Δv is the relativistic time correction.

Some of theses errors can be corrected by additional transmitted information or the use of a two-frequency receiver. However, the clock bias, which does the main damage to the pseudorange measurement cannot be corrected through additional information. That is why equation 2.1 needs to modified as

$$\begin{aligned} \rho_1 &= \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + b_u \\ \rho_2 &= \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + b_u \\ \rho_3 &= \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + b_u \\ \rho_4 &= \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} + b_u, \end{aligned} \tag{2.5}$$

where b_u is the user clock bias error expressed in distance, which is related to the quantity b_{ut} by $b_u = c \cdot b_{ut}$. In Equation 2.5, four equations are needed to solve for unknowns x_u, y_u, z_u and b_u. Thus, in a GPS receiver, a minimum of four satellites is required to solve for the user position [Tsu00].

Up to now it was assumed that the position determination is only preformed by the transmitted satellite clocks. However, in order to make a more precise user position determination a carrier phase measurement is implemented in the GNSS receiver. A carrier phase measurement is a range measurement computed from the GPS carrier signal information. The total number of the carrier cycles from the GPS satellites to the user are measured and converted into a range measurement using the carrier wavelength [Kap96]. The receiver cannot determine the number of integer cycles before the signal is acquired (signal acquisition is discussed in Section 2.6). This is referred to as the integer cycle ambiguity. This ambiguity must be resolved before the carrier phase measurement can be used for position computation. It can be represented by Equation 2.6 [Des04]

$$\theta(t) = -\lambda\varphi(t) = \rho(t) + \Delta D - c(\Delta b - b_{ut}) + c(\Delta T + \Delta\ I + v + \Delta v) + \lambda N, \quad (2.6)$$

where $\theta(t)$ is the carrier phase measurement at time t, $\varphi(t)$ is the carrier phase measurement in cycles, λ is the carrier wavelength and N is the integer carrier phase ambiguity. The carrier phase measurement with ambiguity resolved to the correct integer provides a very accurate range measurement and is used to provide centimeter-level position accuracies.

The user's velocity can be computed by the Doppler measurement. The Doppler is a measure of the instantaneous rate of the GPS carrier phase and is the instantaneous Doppler frequency shift of the incoming carrier. The Doppler shift results due to the relative motion between the receiver and the satellite and is obtained from

$$f_d = f_R - f_T = -f_T \frac{(v_s - v_u) \cdot a}{c}, \quad (2.7)$$

where f_R is the received frequency, f_T is the received transmitted signal frequency, v_s is the velocity of the satellite, v_r is the velocity of the user and the dot product $(v_s - v_u) \cdot a$ represents the radial component of the relative velocity vector $(v_s - v_u)$ along the line-of-sight to the satellite [Kap96].

Acquisition and Tracking of the Doppler shift is discussed in Section 2.5.

2.2. Specifics of NAVSTAR GPS

The Global Positioning System consists of three segments: the space segment, the control segment and the user segment.

This section focuses on the specifics of the GPS space segment. Its pendant, the user segment is presented in Section 2.5. The control segment is not discussed in detail

here, because its main purpose is only to monitor and to improve the information distributed by the satellites, by periodically uploading the prediction of the future satellite positions and satellite clock corrections.

The GPS space segment consists of 32 satellites arranged in six orbital planes with four satellites per plane. The satellite orbit radius is about $26,560\ km$ above the earth center. The satellites have a velocity of about $3.87\ km/s$. Hence a satellite is once every $23h\ 55min\ 56.6s$ above the same point on earth. A satellite has a lifetime of about $7.5\ years$. So in order to avoid failures there are always more satellites as needed in orbit, so that a damaged satellite can be replaced within a short time.

Presently the GPS signals are transmitted on two radio frequencies in the UHF band, also known as the L-Band. These frequencies are referred to as L1 and L2 and are generated as an integer multiple of the $f_0 = 10.23\ MHz$ satellite clock. They are located at

$$f_{L1} = 154 f_0 = 1575.42\ MHz \tag{2.8}$$
$$f_{L2} = 120 f_0 = 1227.60\ MHz. \tag{2.9}$$

In 2013 there will be another frequency band, the L5-Band with $f_{L5} = 1176.45\ MHz$ which is for the Safety-of-Life-Applications and will present the user a high accuracy in his position determination [Tsu00].

The technical description of the GPS signals is given in Section 2.3.

2.3. Signal Structure, Modulation and C/A Code Format

As mentioned in the last section, the GPS satellites transmit on two different frequencies of the L-band, called L1 and L2 frequencies. The transmission scheme used is BPSK DS spread spectrum. Therefore the two carrier frequencies are modulated by a 50 Hz navigation message and a unique *Pseudorandom noise* PRN spreading code, different for each satellite.

Hence the GPS is a *Code Division Multiple Access* (CDMA) system, all satellites use the same frequency band. In order to distinguish the satellites, one must make use of particular PRN codes, supplying certain correlation properties. In GPS there are two spreading codes used, the *coarse/acquisition-code* (C/A-code) and the *precision (encrypted) code* (P(Y)-code). Both of these spreading codes are selected from a

family of Gold codes [Kap96], to maintain a minimum cross correlation. The C/A and P(Y)-codes are in phase quadrature to each other on the L1 frequency; on the L2 frequency there is the P(Y) code only. The chip waveform for both code types equals a rectangular pulse $rect(t/T)$ of duration T respectively to the code rate. The C/A-code is 1023 bits long and has a transmission rate of 1.023 $MChips/s$ (i.e. it repeats every 1 ms and a single C/A-code chip is about 1 μs). As it is not encrypted it is available to civilian users. The P(Y)-code has a length of $6.1871 \cdot 10^{12}$ chips and is transmitted at a rate of 10.23 $MChips/s$ (i.e. it needs one week to be transmitted in total). Further it is encrypted with the Y-code so it is reserved for military applications. The generation of the C/A code is sketched in Figure 2.4. The C/A code generator contains two shift registers known as G_1 and G_2. These shift registers have 10 cells generating sequences with a length of 1023, respectively. The two resulting 1023 chip-long sequences are modulo-2 added to generate a 1023 chip-long C/A-code. To make different C/A-codes for the satellites, the output of the two shift registers are combined in a very special manner. The G_1 register always supplies its output, but the G_2 registers drivers two of its states to a modulo-2 adder to generate its output. The selection of the states for the modulo-2 adder is called the phase selection. By making different phase selections, different codes are obtained for different satellites [Bor07].

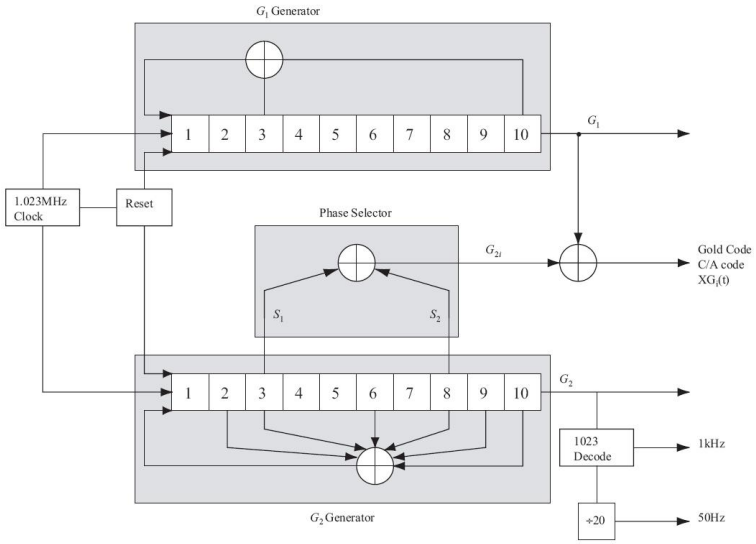

Figure 2.4.: C/A code generator [Bor07].

The basic format of the navigation data is a 1500 *bit* long frame containing five subframes, each having a length of 300 bits. One subframe has a duration of 6 seconds and contains 10 words, with have a length of 30 bits.
The subframes one to three contain the ephemeris data (satellite position and velocity) and are repeated in each frame (i.e. every 30 s), while subframe four and five contain the almanac data, which is repeated every 25 frames (i.e. one entire

navigation message lasts 12.5 *min*).
Furthermore each word in each subframe contains 6 bit for parity check. The first two words of each subframe are the *telemetry* (TLM) and *handover words* (HOW). Figure 2.5 shows the architecture of the navigation message.

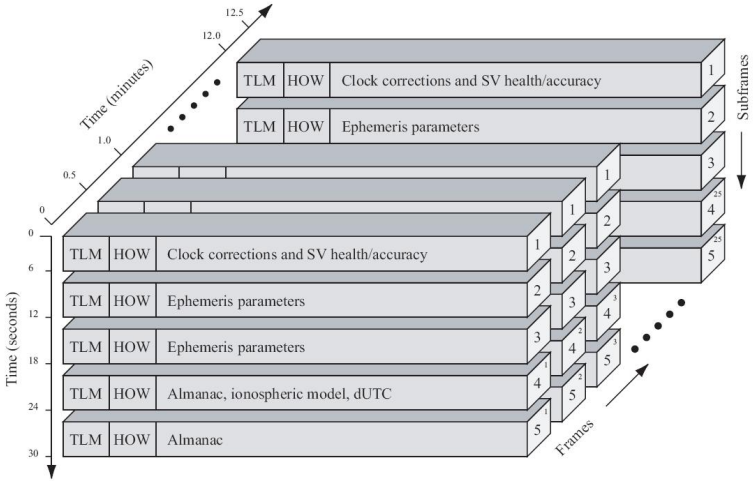

Figure 2.5.: GPS navigation data structure [Bor07].

Last but not least Figure 2.6 shows a block diagram of the GPS satellite transmitter unit. The GPS satellite uses a 10.23 MHz reference clock to generate both the L1 and L2 frequencies. The GPS signal broadcasted is given in Equation 2.10 and 2.11 [Kap96].

$$L_1(t) = \sqrt{2P_{C/A}}C_{C/A}(t)N(t)cos(2\pi f_{L1}t + \phi) + \sqrt{2P_P}C_P(t)N(t)sin(2\pi f_{L1}t + \phi) \tag{2.10}$$

$$L_2(t) = \sqrt{2P_P}C_P(t)N(t)cos(2\pi f_{L2}t + \phi), \tag{2.11}$$

where $P_{C/A}$ and P_P are the C/A and P code signal powers, respectively. $C_{C/A}(t)$ and $C_P(t)$ are the C/A and P codes, respectively; $N(t)$ is the navigation data; f_{L1} is the L1 and f_{L2} is the L2 carrier frequency; ϕ is the initial phase; $L_1(t)$ and $L_2(t)$ are the modulated L_1 and L_2 signals.

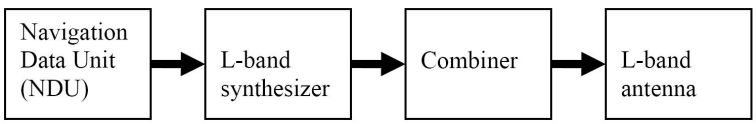

Figure 2.6.: GPS satellite transmitter unit [Des04].

The *navigation data unit* (NDU) generates the cosine and sine of the carrier frequency which are modulated by a 50 Hz navigation data signal. This modulated

signal is spread using the C/A-code and the P(Y) code. The NDU block performs the function of signal modulation, and the synthesizer is used to manipulate the signals according to the C/A-code and the P(Y)-code onto one signal. Both the L1 and the L2 signal are transmitted to the Earth using an L-band antenna [Des04].

Finally, to get an insight in the spread spectrum signal characteristics, Figure 2.7 shows the GPS signal power spectral density. One can see that the PRN codes determine the channel bandwidths required to transmit and receive the spread spectrum signal [Die00].

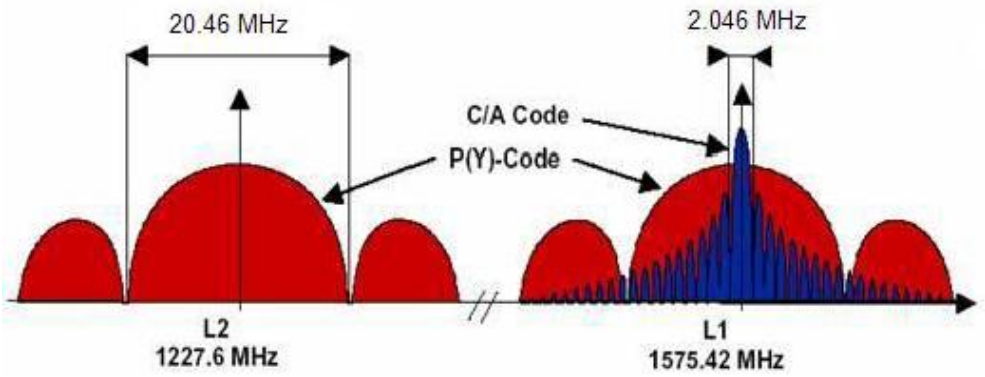

Figure 2.7.: GPS signal power spectral density [Des04].

2.4. Power Levels, Auto-Correlation Function and Power Spectral densities

The received GPS signal power is assumed to be around $-160\ dBW$ for the C/A-code and $-163\ dBW$ for the P(Y)-code in the L1 frequency band and about $-166\ dBW$ for the P(Y)-code in the L2 frequency band. But depending on the user's evaluation angle there are also some variations in the received minimum signal power shown in Figure 2.8.

The received signal powers are below the noise floor of $-140\ dBW/MHz$. Therefore, the GNSS signals will only be detected by receivers using replicated PRN sequences to de-spread the signal.

But before proceeding with a detailed discussion of the correlation process to obtain the de-spread signal, the PRN code spectral properties in the RF domain are examined first.

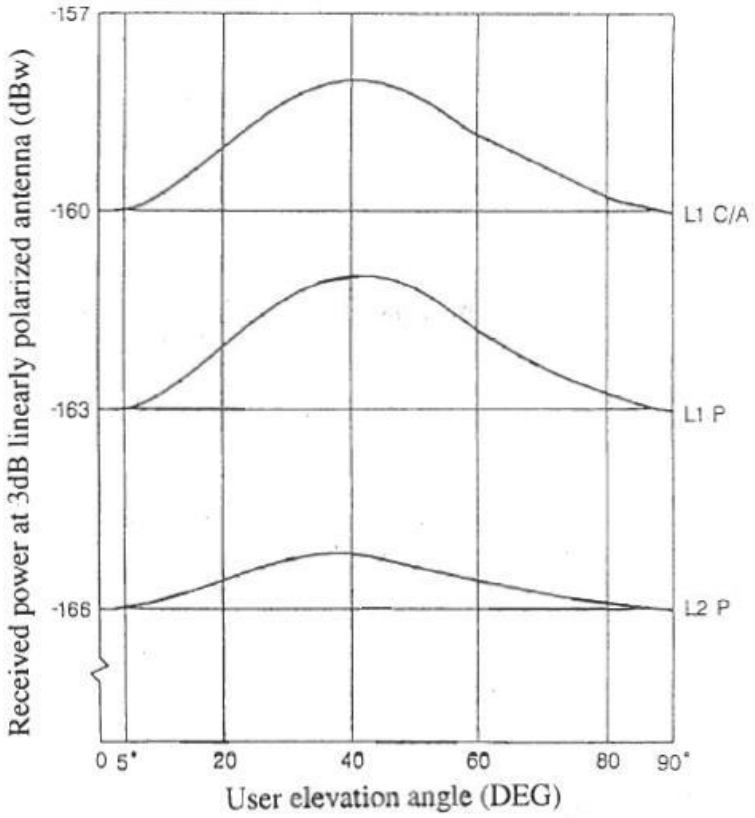

Figure 2.8.: Users received minimum signal power levels as a parameter of the evaluation [Kap96].

The P(Y) code's spectral density function is given with Equation 2.12

$$S_p(f) = P_p T_{CP} \frac{sin^2(\pi f T_{CP})}{(\pi f T_{CP})^2}, -\infty < f < \infty, \tag{2.12}$$

where T_{CP} is the P(Y) code chip with and P_p the carrier power.

The C/A code is, as already mentioned in Section 2.3, a short 1 ms repeating code, thus it has a line spectrum with its components c_j, $j = -\infty$ to $+\infty$ 1 kHz apart. The carrier power is given b$< \sum_{j=-\infty}^{+\infty} c_j = P_c$, the C/A code chip rate is $T_{C/A}$ and the envelope of the line spectrum is

$$S_c(f) = 1000 P_c T_{C/A} \frac{sin^2(\pi f T_{C/A})}{(\pi f T_{C/A})^2}, -\infty < f < \infty, \tag{2.13}$$

Since correlation between incoming signal and code replica is performed, the auto-correlation characteristics of the PRN code are fundamental to the signal demodulation process. Hence, as mentioned in Section 2.3, the PRN codes used are periodic,

predictable, reproducible and have a minimum cross-correlation. Thus these codes are not truly random binary codes, thats why they are called “pseudo” random noise sequences.

The *auto-correlation function* (ACF) of the GPS C/A-code is

$$R_{C/A}(\tau) = \frac{1}{1023 T_{C/A}} \int_0^{1023} C_{C/A_i}(t) C_{C/A_i}(t+\tau) dt, \tag{2.14}$$

where $C_{C/A_i}(t)$ is the C/A-code as a function of time for satellite i, $T_{C/A}$ is the C/A code chip rate (977.5 $nsec$) and τ is the phase of the time shift in the ACF.

The C/A-code's auto-correlation function is a series of triangles with a period of 1023 C/A-code chips or 1 $msec$. Within the intervals between the maximum correlation peaks are also some small correlation peaks . These small fluctuations result from the deviation of line spectrum of the C/A-code which has a $sin(x)/x$ envelope. Figure 2.9 the correlation and spectral characteristics of the C/A-code.

The ACF of the GPS P(Y)-code is

$$R_P(\tau) = \frac{1}{6.1871 \cdot 10^{12} T_{CP}} \int_0^{6.1871 \cdot 10^{12}} C_{P_i}(t) C_{P_i}(t+\tau) dt, \tag{2.15}$$

where $C_{P_i}(t)$ is the P(Y)-code as a function of time for satellite i, T_{CP} is the P(Y) code chip rate (9.78 ns) and τ is the phase of the time shift in the ACF.

The P(Y)-code auto-correlation characteristics are essentially ideal, because its period is so long and its chipping rate so fast. Therefore, the P(Y)-code can be considered essentially uncorrelated with itself [Kap96], see Figure 2.10.

Finally Figure 2.11 shows the correlation gain. One can see that the signal former to the correlation is about 20 dBW below the background noise. But after the gain due to the correlation the signal is approximately 22 dBW above it.

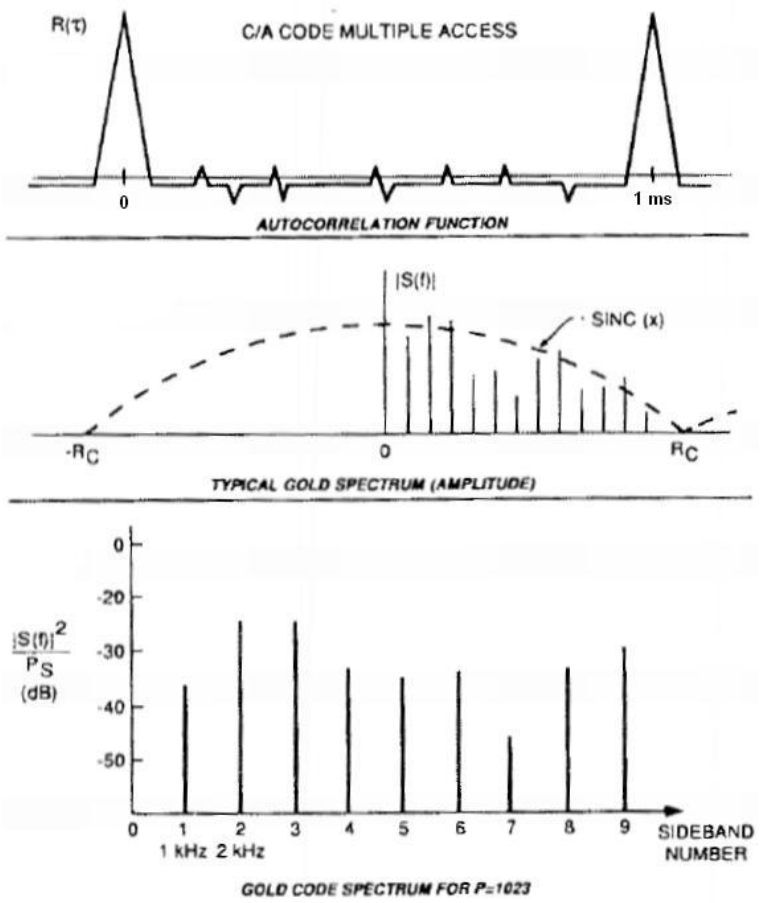

Figure 2.9.: Top: ACF, middle: spectrum and bottom power ratio of the GPS C/A-code [BWP96].

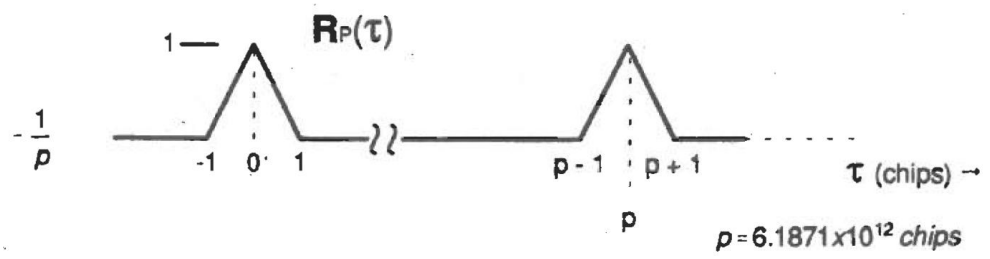

Figure 2.10.: Normalized and simplified ACF of a P(Y)-code with τ in chips [Kap96].

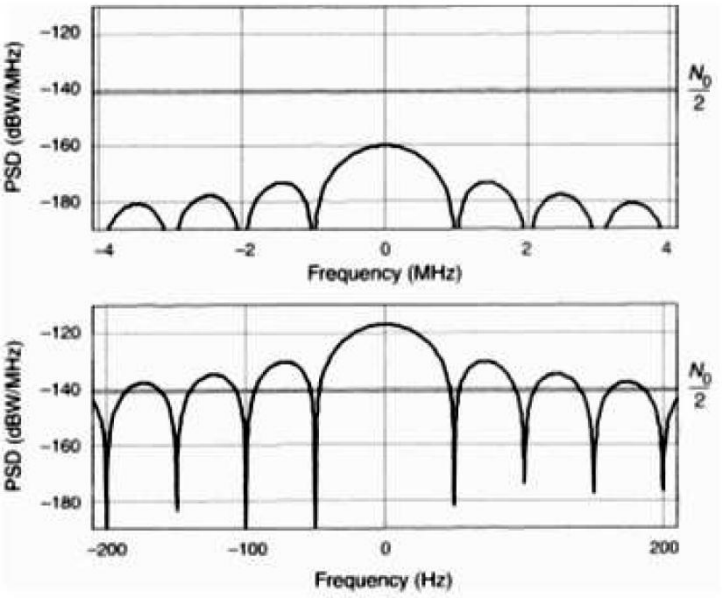

Figure 2.11.: Power spectral densities for the GPS signal and background noise. The top trace shows the power spectral density of the incoming GPS signal, which is below the noise floor. The bottom trace shows the signal after correlation/despreading [Eng06].

2.5. GNSS Receiver Architecture

The structure of a GNSS receiver can be divided into two central parts depending, on the way in which signal processing is performed. Figure 2.12 shows a block diagram of the essential receiver parts.

The first part is called "pre-correlation techniques" due to the fact that here is made no use of the correlation properties. Furthermore this part can be divided into 1. antenna, 2. preamplifier and RF front end, 3. *intermediate frequency* (IF) down conversion and adaptive Analog-to-Digital Conversion, and 4. digital preprocessing. In this work the design of the adaptive ADC and the subsequent digital processing unit are of major concern. Hence Chapter 3 gives a detailed description of how such an adaptive ADC may be implemented and Chapter 6 includes interference mitigation techniques former to the correlation.

In the second part of the GNSS receiver, the navigation information is obtained by performing the correlation. Now signal processing depends on the despread signal, which gives the name of this part: the "post-correlation techniques". Further this part may be divided into 5. correlation, where there is made use of code replica to despread the signal and to obtain in 6. the navigation information so that finally the user's position, velocity and time can be estimated.

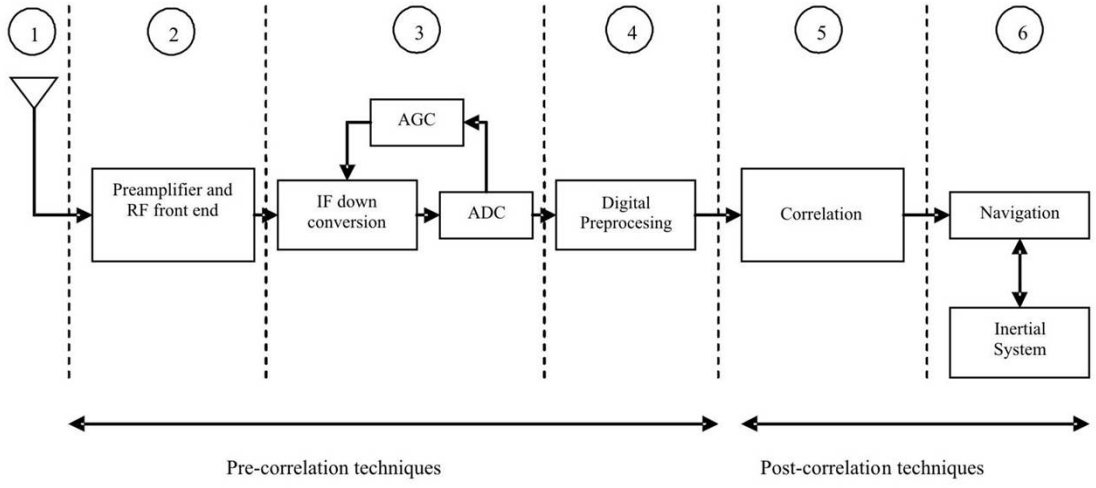

Figure 2.12.: GNSS Receiver Architecture [Web08b].

1. The Antenna is *right-hand circular polarized* (RHCP). It may consist of one or more elements and associated control electronics, and may be passive or active

depending upon its performance requirements. Its function is to receive the GNSS *radiofrequency* (RF) signals of all satellites in view, while rejecting multipath and, if so designed, interference signals through beamforming.

2. The Preamplifier and RF front end consist of burnout protection, filtering and a *low-noise amplifier* (LNA). Its primary objective is to set the receiver's noise figure and to reject out-of-band interference.

3. In the IF down conversion and adaptive ADC part the former RF signal is downconverted to an easier to process IF, by using a super heterodyne architecture. The requirements on the IF section with its AGC are as follows [BWP96]:

(a) Final rejection of out-of-band interference, unwanted sidebands, *local oscillator* (LO) feedtrough, and harmonics. The rejection bandwidth is a trade-off against correlation loss caused by filtering. Further the rejection of wide-band noise is required to minimize aliasing in the sampling circuit.

(b) Increase of the amplitude of the received signal to workable levels for signal processing.

(c) Adaption of the amplification gain to ensure that the incoming signal is spread over all ADC quantization levels.

4. The Receiver's Digital Preprocessing Unit is beside the AGC of major concern in this work. In it pre-correlation digital data pre-processing is performed. This includes the detection of interference and the reduction of its impact on subsequent signal processing units.

Different methods of interference mitigation may be implemented here, such as adaptive Notched filters, frequency excision techniques or temporal blanking. All of them improve the GNSS signal condition and increase the receiver's robustness dealing with interference. Possible realizations of this unit are to find in Chapter 6.

5. In the Correlation Unit of the GNSS receiver acquisition, tracking and decoding of the signal is performed. An acquisition method must be used to detect the presence of the signal. Once the signal is detected, the necessary Doppler and codephase informations are passed to a tracking program. For a description of signal acquisition and tracking see next section.

6. Navigation: After the signal is finally acquired and tracked the user can receive the transmitted information. With this information and knowledge about current code phase and Doppler frequency shift one can set the receivers reference clock and compute the user position and velocity.

2.6. Acquisition and Tracking

In order to acquire, track and decode the GPS signal, the receiver needs to replicate the GPS signal with the different C/A codes first and then to correlate them with the incoming signal. This correlation process yields various peaks that are compared with a detection threshold to test for acquisition success. The replica signal must match the incoming signal both in code and Doppler. The code phase varies due to the range change between the satellite and the receiver. Doppler variation occurs due to the relative motion between the satellite and the receiver [Kap96]. The role of the acquisition is to provide a coarse estimate of the code phase τ and the Doppler frequency f_D to the tracking loops. Therefore it is necessary to know about which frequency range a Doppler shift can occur as the consequence of satellite and user velocity, as it determines the frequency searching range [Des04].

From the orbit speed, the radius of earth and satellite orbit one can calculate the maximum Doppler velocity according to

$$v_d = \frac{v_s r_e}{r_s} = \frac{3874 \cdot 6368}{26560} \approx 929 \frac{m}{s}, \tag{2.16}$$

which is equivalent to the speed of a high-speed aircraft. For the L1 frequency, which is modulated with the C/A code, the maximum Doppler frequency shift is computed as follows

$$f_d = \frac{f_T v_d}{c} = \frac{1575.42 \cdot 929}{3 \cdot 10^8} \approx 4.9 KHz. \tag{2.17}$$

So, for a stationary observer, the maximum Doppler frequency shift is around $\pm 5 KHz$. If the GPS receiver is used in a high-speed vehicle, it is reasonable to assume that the vehicle creates a Doppler frequency shift of about $\pm 5 KHz$ alone, so a maximum Doppler shift of $\pm 10\ KHz$ is possible. These values determine the frequency searching range in the acquisition stage [Tsu00].

The code phase search range extends from 1 to 1023 chips (for the C/A-code). The acquisition process searches the signal for a particular value of the code phase and Doppler frequency over a certain period of time called the predetection integration time. The acquisition time is determined by the predetection integration period

and the number of cells (obtained from code phase and Doppler range) to search. The GPS receiver can compute visible satellites from approximate knowledge of the receiver position, the GPS time and the almanac data which reduces the number of satellites to be searched and speeds up the time-to-first-fix [Des04].

The Time Domain Correlation Technique is the conventional method for acquisition. It is a two-dimensional search in time (code phase) and frequency. The search range is therefore divided into cells, wherein each cell (search bin) represents a particular code delay and Doppler frequency, see Figure 2.13.

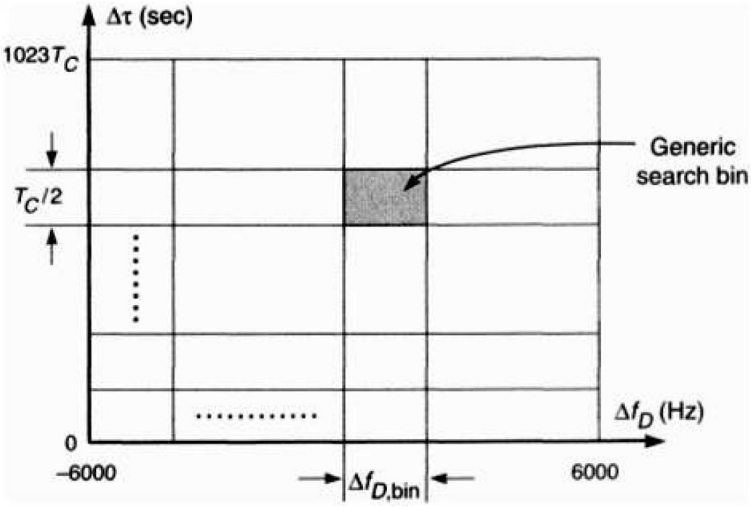

Figure 2.13.: Signal search area covers Doppler frequency and code phase [Eng06].

Correlation is performed in each cell for the predetection integration period. The obtained correlation value is compared against a detection threshold by performing a hypothesis test. If it exceeds that threshold, the satellite is declared as acquired otherwise the search is continued into the next cell. The total number of cells to be searched is given by the number of code delay cells times the number of Doppler bins. Figure 2.14 shows a typical C/A-code signal power profile about the signal's main correlation peak.The peak location is related to a C/A-code phase and a frequency of the signal.

After the GPS signal is declared to be acquired, the receiver changes its mode to tracking, in order to keep lock on the code phase and Doppler shift. The preceding acquisition process provided coarse estimates of the code phase and the Doppler frequency. Doppler tracking is performed in a *Phase-Locked-Loop* (PLL) and code tracking in a *Delay-Locked-Loop* (DLL). The idea behind the DLL is to correlate the input signal with three replicas of the code seen in Figure 2.15.
The first step in Figure 2.15 is converting the C/A code to baseband, by multiplying the incoming signal with a perfectly aligned local replica of the carrier wave.

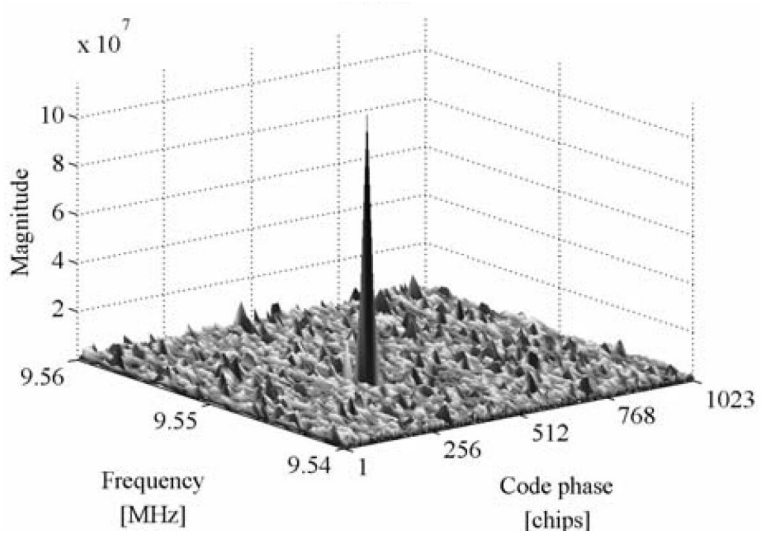

Figure 2.14.: Two-dimensional search in code phase and frequency [Bor07].

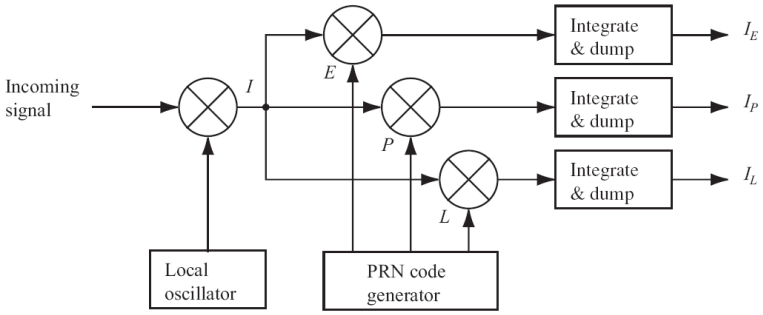

Figure 2.15.: Basic code tracking loop block diagram [Bor07].

Afterwards the signal is multiplied with three code replicas. The three replicas are nominally generated with a spacing of $\pm\frac{1}{2}$ chip. After this second multiplication, the three outputs are integrated and dumped. The output of these integrations is a numerical value indicating how much the specific code replica correlates with the code in the incoming signal. The three correlation outputs I_E , I_P, and I_L are then compared to see which one provides the highest correlation.

Figure 2.16 shows an example of code tracking.

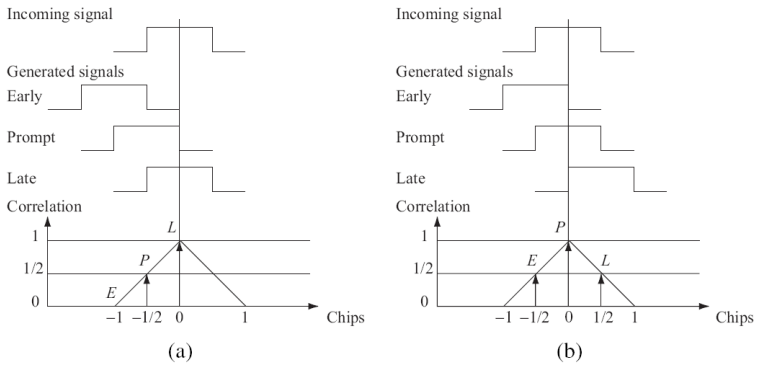

Figure 2.16.: Code tracking. Three local codes are generated and correlated with the incoming signal. (a) The late replica has the highest correlation so the code phase must be decreased, i.e., the code sequence must be delayed. (b) The prompt code has the highest correlation, and the early and late have similar correlation. The loop is perfectly tuned in. [Bor07].

In Figure 2.16a the late code has the highest correlation, so the code phase must be decreased. In Figure 2.16b the highest peak is located at the prompt replica, and the early and late replicas have equal correlation. In this case, the code phase is properly tracked [Bor07].

Another parameter, jet to discuss, is the achieved *carrier-to-noise ratio* (C/N_0) at the correlator output, which is directly related to the tracking. Because estimating C/N_0 is very similar to code lock detection.

In this work the achievd C/N_0 is used to evaluated the performance of the developed interference mitigation techniques. The C/N_0 can be determined by comparing the total signal-plus-noise power in two different bandwidths. Therefore the total power is measured in a $1/T$ noise bandwidth (wide-band power) before the correlation and in a $1/MT$ noise bandwidth (narrow-band power) after the correlation, of the

following form

$$WBP_k = \left(\sum_{i=1}^{M} (I_i^2 + Q_i^2) \right)_k \tag{2.18}$$

and

$$NBP_k = \sum_{i=1}^{M} (I_i * C_{C/A}(i+k))^2 + \sum_{i=1}^{M} (Q_i * C_{C/A}(i+k))^2 \tag{2.19}$$

computed over the same M samples and "$*$" representing the convolution. So for an unknown post correlation noise power the exact relationship between these measurements and signal-plus-noise power is not known. However, a normalized power can be defined as

$$NP_k = \frac{NBP_k}{WBP_k} \tag{2.20}$$

which obtains statistics that provide a monotonic function of C/N_0.

After the correlation process one can compute the lock detector measurement as follows

$$\widehat{\mu}_{NP} = \frac{1}{K} \sum_{k=1}^{K} NP_k \tag{2.21}$$

where K is a parameter to reduce the standard deviation by a factor of $\sqrt{K}$.

Finally one is able to compute with Equation 2.22 the desired C/N_0,

$$\frac{C}{N_0} = 10log_{10} \left(\frac{1}{T} \frac{\widehat{\mu}_{NP} - 1}{M - \widehat{\mu}_{NP}} \right), \tag{2.22}$$

and so asses the quality of the received signal and the performance of the developed interference mitigation circuits [BWP96].

3. Automatic Gain Control for a Multibit Analog-to-Digital Converter

Modern GNSS receivers are composed of an analog front-end and a digital part that involves, amongst others, code and carrier tracking circuits, as shown in Section 2.5. The conversion from the analog state to the digital domain is done by an ADC, its function is to sample and quantize the incoming signal.

In many of the nowadays available commercial GPS receivers only 1-bit quantization (hard limiting) takes place. But whenever it comes to *safety-of-life* (sol) applications a multibit ADC is necessary to reduce quantization losses and to supply further digital processing units with a high signal resolution to improve the receiver's robustness. For a 1-bit ADC in the presence of Gaussian noise the SNR is degraded by 1.96 dB through the quantization, whereas in CW interference the degradation is much greater. But even if there is no interference, a value 1.96 dB for the quantization loss is only true for infinite bandwidth and infinite sampling frequency, otherwise it is larger. For instance, in a bandpass hard limiter the loss of SNR is well known to be 6 dB. Such high losses are not acceptable for sol-applications. Hence multibit ADCs are used, where for instance, the quantization loss for a 2-bit ADC, with an adaptive quantization threshold, is less than 0.6 dB in the presence of Gaussian interference.

Due to the fact that normally the quantization thresholds are fixed in an ADC, an AGC is necessary to adjust the incoming signal in relation to its variance and the maximum ADC quantization thresholds. This chapter presents the fundamentals of multibit A/D-conversion and different AGC concepts to adjust the ADC input power in relation to its maximum quantization threshold to guarantee minimum quantization losses.

3.1. Multibit Analog-to-Digital Conversion

In order to quantize a signal with more than 1-bit resolution, different quantization laws can be used. The most prevalent ones are uniform although nonuniform laws have been proposed, for example to mitigate CW interference in [Amo83] as shown Chapter 6. Quantization laws may be non-centered so that there is no zero level. A possible quantization law is given below

$$X_Q(X) = \begin{cases} (M-\frac{1}{2})\Delta & \text{if} \quad M\Delta < X \\ (m-\frac{1}{2})\Delta & \text{if} \quad (m-1)\Delta < X \leq m\Delta \quad m \in 1..., M \\ (m+\frac{1}{2})\Delta & \text{if} \quad m\Delta < X \leq (m+1)\Delta \quad m \in 1..., M \\ (-M+\frac{1}{2})\Delta & \text{if} \quad X \leq -M\Delta \end{cases}, \tag{3.1}$$

where X is the incoming signal and the quantized signal is X_Q, the number of levels is $M = 2^n$, n being the number of bits used and Δ is the quantization step. All the possible 2^n states are used so it is an even state converter without zero state. Saturation effects appear when the absolute signal amplitude is above $M\Delta$. Figure 3.1 presents a uniform 2-bit non-centered law, where the output signal takes value in the following bins: -3, -1, $+1$ or $+3$. The input sampled signal X corresponds to the x-axis while the output quantized signal X_Q is presented on the y-axis. L is called the maximum quantization threshold.

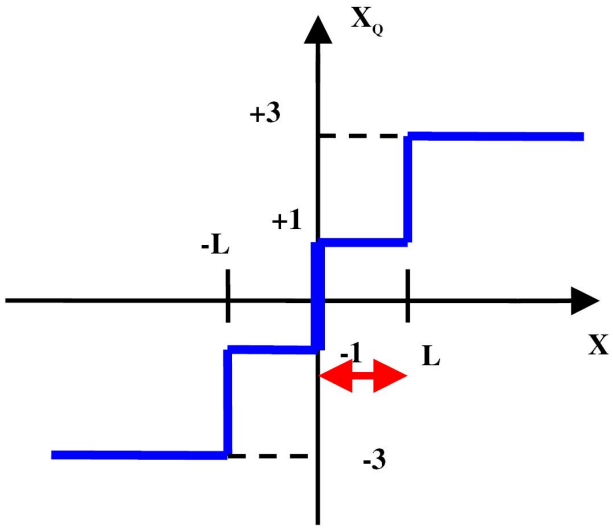

Figure 3.1.: 2-bit non-centered and uniform quantization law [Bas03].

Through the quantization process an additional error is introduced to the quantized signal, often considered as an additive noise n_Q:

$$X_Q = X + n_Q \tag{3.2}$$

This noise is assumed to be white and Gaussian. It induces small SNR degradations, which can be calculated from the quantization law, the number of bits used and the signal's standard deviation. Figure 3.2 shows the SNR degradation at the correlator output assuming only white Gaussian thermal noise at the ADC input, neglecting the pre-correlation filtering and the limited sampling frequency, for ADC having 1 to 5 bits.

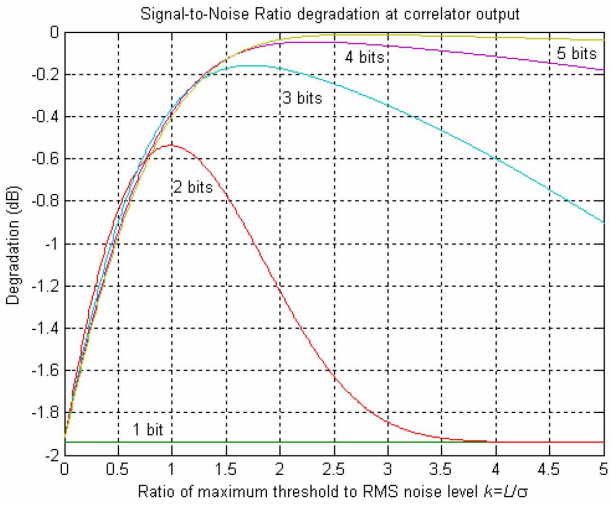

Figure 3.2.: SNR degradation at correlator output in presence of thermal noise only for multiple bits. Pre-correlation filtering and finite sampling frequency effects are neglected [Bas04].

The presented SNR degradation depends on the ratio $k = L/\sigma$, where L is the maximum quantization threshold (saturation level), and σ is the input signal's standard deviation. Surprisingly this ratio does not depend on the GNSS signals themselves, as they are completely covered in noise.
Further, the degradation depends also on the number of ADC bits, i.e. the number of thresholds in a ADC. As one can see, for each curve, there is an optimal ratio k, for which the quantization loss is minimal. These minimum degradations for different number of bits are summarized in Table 3.1. The theoretical derivation for the minimum quantization degradation loss is presented in [Spi77, Page 552].

	1 bit	**2 bit**	**3 bit**	**4 bit**	**5 bit**	**>5 bit**
Optimal k_{opt}	—	0.9890	1.7310	2.2910	2.7225	3
Minimum degradation (dB)	1.96	0.5369	0.1589	0.0472	0.0138	< 0.0138

Table 3.1.: Minimum SNR degradation at correlator output due to quantization and associated optimal ratio. Pre-correlation filtering and finite sampling frequency effects are neglected [Bas04].

3.2. A/D-converters with a Dynamic Range

This Section examines how to adjust the total available gain according the total ADC range, when the ADC has more bits than required for further digital processing in the receiver. This is a straightforward extension when compared with the classic design as presented in Section 3.1.

Assume that bit_{init} bits (for instance 2) are initially used in digital tracking loops but a bit_{total}-bit ADC (for instance a 5 *bit* ADC) is available. Section 3.1 showed that there is an optimal ratio k_{opt} that minimizes SNR degradation at correlator output. Assume the maximum quantization level L is constant so there is an optimal incoming noise standard deviation σ_{opt} which depends on both L and the ratio k. In that case, the maximum quantization threshold equates

$$L = \left(2^{bit_{init}-1} - 1\right)\Delta. \tag{3.3}$$

The quantization step is denoted Δ and the partition that represents the limits between quantization intervals is

$$P = -L : \Delta : L. \tag{3.4}$$

The additional bits, $bit_{total} - bit_{init}$ in total, may be used either to increase the resolution or to increase the dynamic range. The dynamic range is defined as twice the maximum signal amplitude for which the ADC is no saturated. Assume bit_{res} additional bits are dedicated to increase the resolution then the new quantization step is

$$\Delta_{new} = \frac{\Delta}{2^{bit_{res}}}. \tag{3.5}$$

The numbers of bits must, of course, comply with this relationship

$$bit_{init} + bit_{res} \leq bit_{total}. \tag{3.6}$$

The new maximum quantization level is

$$L_{new} = \left(2^{bit_{total}-1} - 1\right)\Delta_{new} \tag{3.7}$$

and the new partition is

$$P_{new} = -L_{new} : \Delta_{new} : L_{new}. \tag{3.8}$$

So as to obtain samples on bit_{init} bits, quantized samples on bit_{total} must be grouped according to the initial partition P. Figure 3.3, shows the ADC bins loading, in presence of Gaussian thermal noise only, if 2 bits are used in digital tracking loops ($k_{opt} = 0.98$) but that 5 bits are available in total. Quantized values are between -31

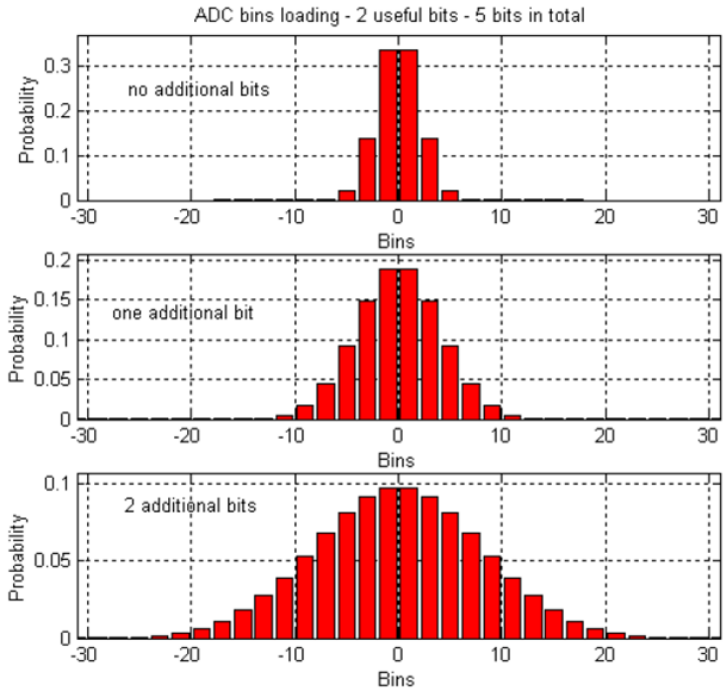

Figure 3.3.: ADC bins loading for 2 useful bits but 5 bits in total with several additional bits (0, 1 and 2) to increase resolution [Bas04].

and +31, by steps of 2, to ensure a uniform and non-centered quantization. Subplots correspond, respectively, to the absence of additional bits, 1 and 2 additional bits to increase the resolution.
Additional bits may also be used to increase the dynamic range of the AGC/ADC system. Assume the nominal bin distribution is chosen to be narrow over the full range as shown on the first subplot of previous figure. Then if a strong interference, such as a CW, appears the AGC gain will decrease to ensure the optimal ratio k. However the variable analog gain has a limited range such that the AGC gain cannot decrease below a certain fixed value. It may imply saturation in traditional AGC/ADC implementations. Now using more bits, bins distribution may be spread out over the full range of bins preventing the quantization process from saturation. The global AGC/ADC gain range is then larger. Of course it means that the ADC must also be adaptive and the interference detection thresholds, as presented in Chapter 5, will not be constant. The dynamic range of the ADC using previous methodology is

$$2^{bit_{total}-1}\Delta_{new} = 2^{bit_{total}-1} \cdot \frac{\Delta}{2^{bit_{res}}} = 2^{bit_{total}-bit_{res}-1}\Delta. \tag{3.9}$$

Clearly, dynamic range decreases as the selected resolution is better. Figure 3.4 illustrates the quantization characteristics of the ADC as a function of the additional bits used to increase ADC resolution. The x-axis corresponds the input signal values

and the y-axis corresponds to the quantized output values. Thus the x-axis represents the dynamic range of the ADC. The quantized values are again between −31 and +31, by steps of 2. This plot shows how the dynamic range increases when the resolution decreases. Table 3.2 summarizes these results. Again 2 bits are initially necessary in digital tracking loops but 5 bits are available in total. For this illustration, the initial (no additional bits) quantization step Δ set to 1 volt [Bas04].

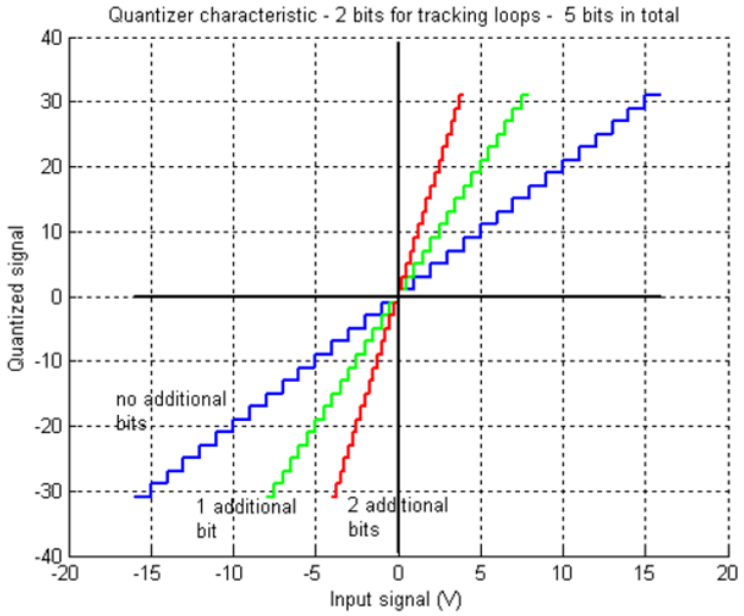

Figure 3.4.: Quantizer characteristic for 5 bits in total (2 bits initially) and with various additional bits for increased resolution [Bas04].

number of additional resolution bits	**0 bit**	**1 bit**	**2 bit**
dynamic range (V)	32	16	8

Table 3.2.: Dynamic range for 0, 1 or 2 additional bits dedicated to resolution increase. [Bas04].

Thus one has the choice of the repartition of the total number of bits. It is a trade-off between ADC resolution and AGC/ADC total dynamic range.

In this thesis a 14 *bit* ADC is used, but only 6 *bit* are necessary for further signal processing, such as digital beamforming and acquisition/tracking of the GNSS signal. The additional 8 bits increase the robustness of the pre-correlation interference detection and mitigation techniques presented in Chapter 5 and 6. As simulations show, using an 6 *bit* initial ADC loading leads to a excellent performance of the

frequency excision techniques presented in Section 6.3, because when strong narrowband interference occurs, so that the signal is attenuated by about 20 *db* through the AGC to avoid saturation effects in the ADC, the GNSS superposed with RFI and thermal noise is still quantized in a high enough resolution (≈ 2.5 *bit*), so that those frequency excision techniques can remove the interference properly. Notice that, as stated at the beginning of this section, samples must be represented in the end on $bit_{init} = 6$ *bit* to fit the specifications of subsequent signal processing system.

3.3. Automatic Gain Control for an optimal A/D-conversion

The examinations of Section 3.1 showed that there is an optimal ration k_{opt} that minimizes quantization losses. Due to the fact that the maximum ADC quantization threshold L is fixed, it is necessary to adjust the incoming thermal noise variance σ in such a way that minimum quantization losses are obtained. Therefore an AGC is implemented. Its primary role is to ensure that the optimal ratio $k = L/\sigma$ is respected. The AGC is placed right before the ADC and is basically an amplifier, of which the variable gain is controlled by a feedback loop. Figure 3.5 shows the basic AGC feedback circuit.

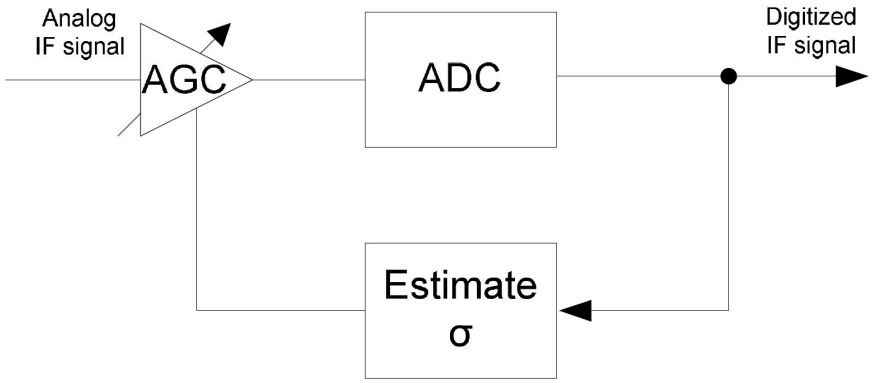

Figure 3.5.: Bais AGC feedback circuit.

Another function of the AGC is to increase the dynamic range. If an interferer, for example a continuous wave like interference appears, it can easily cause saturation in the ADC. By decreasing the gain, the AGC can avoid, or at least reduce these saturation effects, which has the benefit that the receiver can deal with stronger interference powers.

As a forecast, the AGC can be combined with other interference mitigation tech-

niques as shown in Chapter 6. Having the cause that a high robustness of the receiver, even in the presence of strong interference powers, is achieved.

But before proceeding with the AGC overall design, two different AGC implementations strategies are presented first.

The analog realizations of the AGC senses the analog signal to estimate σ and to adjust the gain of an amplification stage accordingly. However, this AGC implementation is not concerned in this thesis as it is analog and further it is neither able to deal with such high sampling rates as they are used here, nor it is convenient when digital blanking shall be implemented to cope with pulsed signals (for a description of pulsed signals see Section 4.1.2).

The digital AGC implementation, is one of the main objectives of this thesis. Here the ADC output samples are used to form statistics, steering the amplifier before the ADC. Related to this, two possible strategies driving AGC are presented in the next sections. The first one is to set the gain in such a way that the ADC bins Gaussian shape is conserved. The second one maps the ADC output power to its input, meaning that one can take conclusion of the current input power on the basis of the estimated ADC output power to steer the AGC.

3.3.1. AGC Regulation through Conservation of the ADC Bins Gaussian Shape

Due to the fact that in the absence of interference thermal Gaussian noise is the prevalent signal, the observable distribution on ADC bins is also of Gaussian shape. So, due to the fact that the optimal ratio $k_{opt} = L/\sigma$ for different ADCs is known from table 3.1, one can compute the optimum bin loading with the following the equation

$$N_{nom}(i) = \frac{1}{2} erfc\left(\frac{b(i)}{\sqrt{2}\sigma_{opt}}\right) - \frac{1}{2} erfc\left(\frac{a(i)}{\sqrt{2}\sigma_{opt}}\right), \tag{3.10}$$

where $N_{nom}(i)$ is the nominal bin loading for the ith bin, when the input variance σ_{opt} is set, $a(i)$ and $b(i)$ are the lower and larger borders of the ith bin and $erfc$ is the complementary error function.

Table 3.3 indicates the optimum bin-loading for a 2-bit ADC and Figure 3.6 shows this optimal bins distribution for uniform non centered 2 bit-quantizer in a histogram.

ADC bins	**-3**	**-1**	**+1**	**+3**
Loading(%)	16.35	33.65	33.65	16.35

Table 3.3.: Optimal ADC bins loading for uniform non centered 2 bit-quantizer [Bas04].

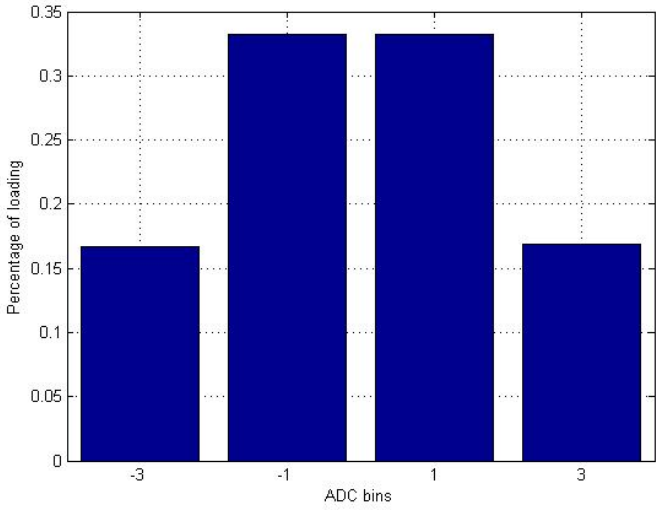

Figure 3.6.: Optimal 2-bits ADC bins loading with $k = 0.986$.

If these bins loadings are obtained, the optimum gain for minimum quantization loss is set.

Now, examining this distribution, it catches one's eye that the difference of loading between bin "-3" and bin "-1" is about 0.18 and the same difference exists between bin "+1" and "+3", so a possible error ϵ can be constructed to drive the AGC feedback loop

$$\epsilon = \left(\frac{N(-1) + N(+1)}{2} - \frac{N(-3) + N(+3)}{2} \right) - 0.18, \tag{3.11}$$

where $N(i)$ is the current ith ADC bin loading.

In this work, that AGC design is not implemented, because there are some issues using this strategy for a more than 2 bit ADC, i.e. one would need to make use of the *Least-Mean-Square* (LMS) algorithm, because at a more than 2 bit quantizers the ADC bins distribution might clearly represent the distribution of the incoming signal, which is for example for a CW-RFI not of Gaussian shape. But anyway, by making use of power mapping techniques, as explained in the next section, an excellent controlling method was found, which did not make it necessary to implement this gain steering strategy, too.

3.3.2. AGC Regulation through Power Mapping

A quite different gain steering approach is to measure the ADC output power or variance, and on the basis of that taking conclusions to estimate the current ADC input power. In [Cho91] a theoretical discussion of estimating the ADC input power on the basis of the ADC output power is given.

So AGC steering can be performed with the estimated ADC output power/variance and a PI-controller or a look-up-table, which is also the most common AGC implementation in a GNSS receiver. The following sections present these AGC types.

AGC using a Look-Up-Table

Using a look-up-table means that the gain cannot take every value, it can only be set to those listed in that table. This leads to the fact that the optimum ratio k_{opt} might not be set. But anyway, this is the most common way to drive an AGC on the basis of the estimated ADC output power/variance, because the achieved ratio $k = L/\sigma$ will be quite close to the optimum one.

This section show how to generate that look-up-table first and second how to use it for varying interference powers.

The look-up-table is generated as follows:

(1) A sample chain, representing the receivers noise floor is generated using the MATLAB "randn"-function.

(2) Than the front-end-gain is set in such a way, that its gain together with the maximum gain of the variable gain amplifier leads to a signal variance that respects the optimum ratio k_{opt} for minimum quantization losses as depicted in Table 3.1.

(3) Next interference is introduced to the noise floor by increasing the power of an additional white Gaussian noise source.

(4) Based on the knowledge of the current interference power the necessary gain to respect the optimal ratio k_{opt} is estimated. The interference power is increased that long, until the gain has decreased to 0 db, thereby for each interference power the according gain is stored in a table.

(5) Last but not least, for every gain value in the table, the ADC output variances are estimated for all possible interference powers before the gain has dropped to 0 dB and are stored also stored in that table.

After the look-up-table is generated one also needs a method to find the best fitting gain in that look-up-table to respect the ratio k_{opt}. The search algorithm works as follows:

(1) According to the current gain, the belonging table with the possible ADC output variances is opened.

(2) Than the current ADC output variance is estimated.

(3) Corresponding to this variance, that table value is picked, which is closest to the value of the current ADC output variance.

(4) Having picked this value the belonging gain is set and the algorithm is restarted.

In order to clarify the table search algorithm, Figure 3.7 illustrates the way it works, as an example for a 6 bit ADC. At the beginning it is assumed, that only noise was present, so the AGC was at its maximum gain, which is 20 dB in this work. Now

a interference source is introduced that leads to a ADC output variance of ≈ 58. According to this ADC output variance, the arrow (a) indicates the corresponding gain value, which the variable gain amplifier has to take in order to keep the input power constant. Finally one measures the ADC output variance again, which is now about 21, as it would also be for the RFI free case and a gain of 20 dB. But as indicated in the description of the table search algorithm, one is now in a different table of possible ADC output variances, so that now 21 is associated with a gain of 10 dB. Therefore the double arrow (b)-(c) indicates that the gain will stay constant as long as the interference power does, else it will change.

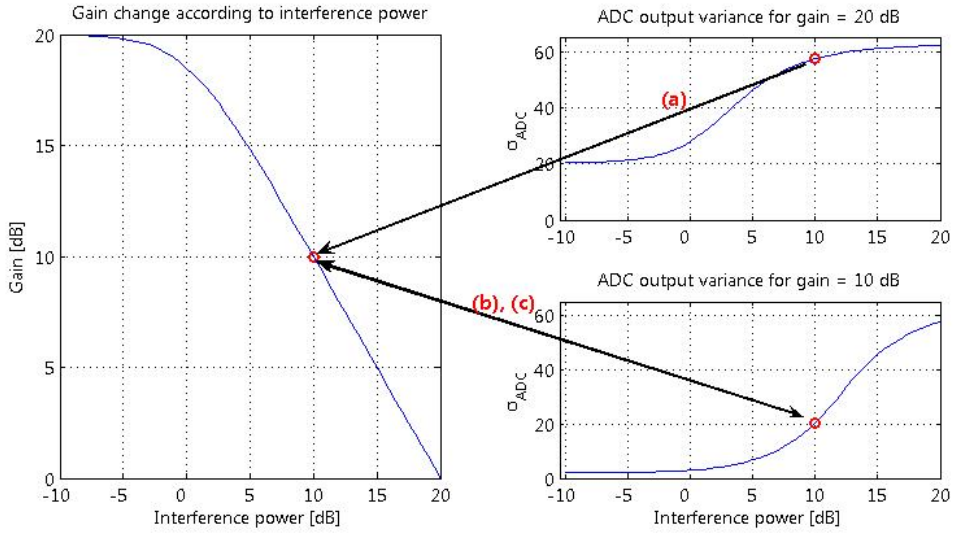

Figure 3.7.: Graphic illustration of the use of a look-up-table.

Finally the AGC over all design, using a look-up-table, is shown in Figure 3.8, but notice: The output of the ADC, i.e. $\mathbf{x}(n)$ is a vector containing quantized samples of a certain timespan. Hence the gain stays constant during this timespan, this is necessary to form statistics of the ADC output variance, so that the table search algorithm can work properly. The necessity of ADC output vectors is even more clarified in Section 5.3, where it comes to the computation of the discrete signal spectrum on the basis of those vectors.

AGC using a PI-controller

As explained in the last subsection, the aim of a AGC is to keep the input power constant, which implies that also the ADC output power stays constant. So there

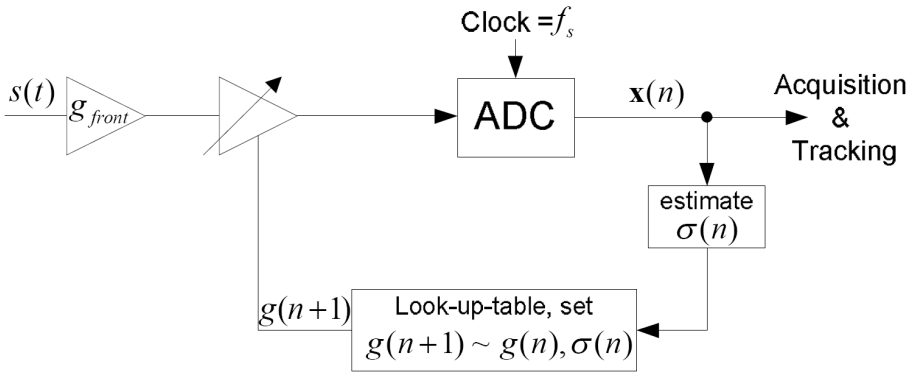

Figure 3.8.: AGC design, using a look-up-table.

is a different gain steering approach, using an error signal, which is defined as

$$\epsilon = \sigma^2_{current} - \sigma^2_{target}, \tag{3.12}$$

where $\sigma^2_{current}$ is the current ADC output power and σ^2_{target} is the ADC output power for the RFI free case.

So with this error signal one can easily adapt the gain with a PI-controller according to equation

$$g(n) = g_{init} + \sum_{k=0}^{n} k_i \epsilon(k), \tag{3.13}$$

where g_{init} is the initial gain, 20 dB in this thesis, n is the current discrete point in time and k_i the integrator time constant. The structure of this AGC design is presented in Figure 3.9.

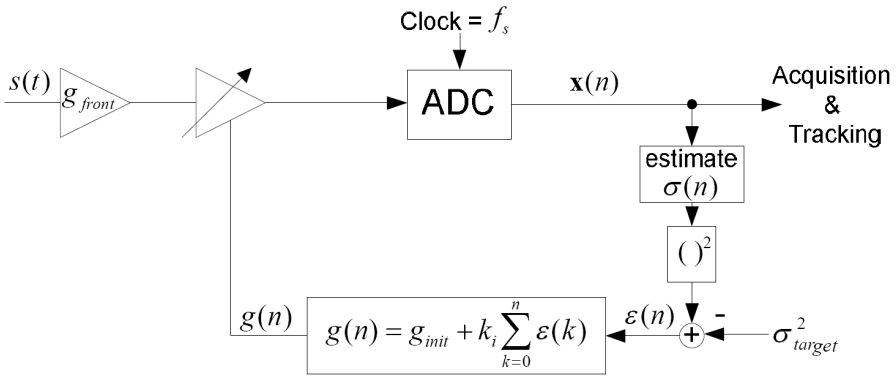

Figure 3.9.: AGC design, using a PI-controller.

This approach works quite fine, for slowly varying interference powers. It will even perform a better gain steering for noise being the dominant signal, because the integrator works as a low pass filter and so is less sensitive of power fluctuations in the received noise floor. But when sudden strong interference powers occur it has great disadvantages against the look-up-table strategy, because it needs some time to reduce the gain in which the GNSS signal might be lost due to saturation effects in the ADC.

Because of this the parameter k_i would need to be quite large, but that causes instabilities in the circuit. Hence only the AGC using a look-up-table is concerned in the following.

3.3.3. AGC Issues in Presence of non Gaussian interference

The AGC design of Figures 3.8 and 3.9 are only appropriate when there is only white Gaussian thermal noise present on the ADC input. In presence of interference three issues are raised:

(1) The AGC functioning is disturbed. It estimates the standard deviation, which is corrupted by interference.

(2) The ADC's input signal does not follow a Gaussian distribution, as explained in Section 3.3.1.

(3) The modification may worse the signal (notably interference) for the correlation process. In order to clarify this, Figure 3.10 shows the SNR degradation, using a 3 bits ADC, at the correlator output, in presence of a CW-RFI in different configurations.
The first curve corresponds to the absence of AGC/ADC, known as the linear case. The second one is in the case of optimal gain adaption and the last one is where only Gaussian thermal noise is assumed to drive the AGC. Thus, for large J/N ratios, the difference between optimal and classic adaption may be large. For example, the result is about 10 dB for $J/N = 20$ dB. Thus a receiver that is intended to mitigate CW simply using the AGC should adaptively change its quantizer interval as shown in Section 6.1 and [Amo83], or as performed in this thesis, one might combine the AGC with more complicated solutions, such as spectral excision, where the main objective of the AGC is only to avoid saturation effects in the ADC [Bas03].

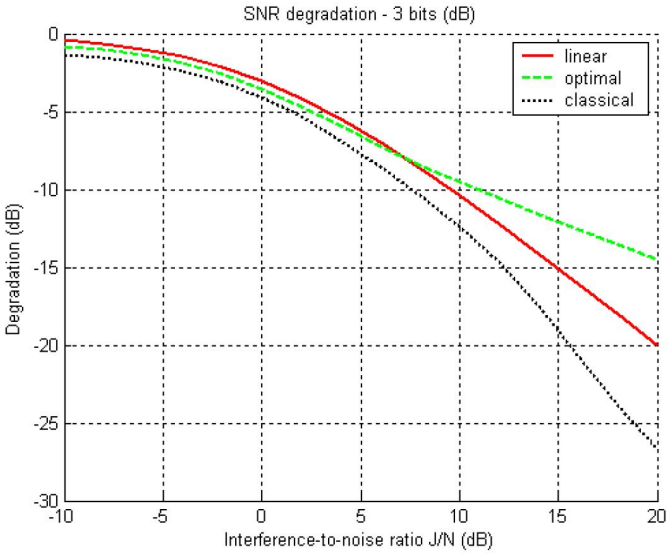

Figure 3.10.: SNR degradation at correlator output in presence of a CW-FRI, for different gain adaption techniques [Bas03].

4. Radio Frequency Interference

The objective of this thesis is to mitigate the impact of *radio-frequency-interference* (RFI) on the correlator. But before any interference detection or mitigation techniques can be developed it is necessary to investigate what types of interference for a receiver operating in the GPS L1 and L5 frequency band may occur.

Therefore this chapter examines various RFI types, their typical sources, as well as their impact on the GNSS receiver. Intentional interference such as jamming is not considered here.

In Section 4.1 different types of RFI and their typical sources are summarized, afterwards their effects on the GNSS Receiver and especially on the ADC are analyzed in Section 4.2.

4.1. Types of Interference

Table 4.1 summarizes different types of RFI and their typical sources. Thereby the main focus lies on the examination of *(Ultra)-Wide Band*, *Pulsed* and *Continuous Wave* interference, as these are the most common interference types corrupting the GNSS signal in the GPS L1 and L5 frequency bands.

4.1.1. Wideband Interference

Wideband Interference, also referred to as *Ultra-Wide Band* (UWB) RFI has a bandwidth that is defined as the frequency band bounded by the points that are 10 dB below the highest radiated emission, as based on the complete transmission system including antenna. It can occur as the transmission of an UWB device that is an intentional radiator that, at any point in time, has a fractional bandwidth equal to or greater than 0.20 or has a UWB bandwidth equal to or greater than 500 MHz, regardless of the fractional bandwidth [Web08a].

Type	Typical Source
Wideband-Gaussian	Intentional noise jammers
Wideband phase/frequency modulation	Television transmitter's harmonics or nearband microwave link transmitters
Wideband-spread spectrum	Intentional spread spectrum jammers or near-field of pseudolites
Wideband pulse	Radar transmitters
Narrowband phase/frequency modulation	AM stations transmitter's harmonics
Narrowband swept continuous wave	Intentional CW jammers or FM stations transmitter's harmonics
Narrowband continuous wave	Intentional CW jammers or near-band unmodulated transmitter's carriers

Table 4.1.: Types of RFI and possible sources [Kap96]

There are two common forms of UWB: one based on sending very short duration pulses to transmit information and another approach using multiple simultaneous carriers. The most common form of multi-carrier modulation, *Orthogonal Frequency Division Multiplexing* (OFDM), has become the leading modulation for high data rate systems. Whereas, the pulsed UWB is more suitable for the use with ranging and measurement systems.

The *multi-carrier UWB* (MC-UWB) systems transmits a signal $s(t)$ that has the following complex baseband form

$$s(t) = A \sum_{r} \sum_{n=1}^{N} b_n^r p(t - rT_P) e^{j2\pi n f_0 (t - rT_P)}, \tag{4.1}$$

where N is the number of subcarriers, b_n^r is the symbol that is transmitted in the rth transmission interval over the nth subcarrier, and A is a constant that controls the transmitted power spectral density and determines the energy per bit. The fundamental frequency is $f_0 = 1/T_0$ [Web08a].

OFDM is a special case of multi-carrier transmission that permits sub-carriers to overlap in frequency without mutual interference and hence spectral efficiency is increased. Multiple users can be supported by allocating each user a group of sub-carriers. OFDM-UWB is a novel system that has been proposed as a physical layer for high bit rate, short-range communication networks. OFDM-UWB is one proposed physical layer standard for 802.15.3a Wireless Personal Area Networks. OFDM-UWB uses a frequency coded pulse train as a shaping signal. The frequency

coded pulse train is defined by

$$p(t) = \sum_{n=1}^{N} z(t - nT) e^{-j2\pi c(n)\frac{1}{T_c}}, \tag{4.2}$$

where $z(t)$ is an elementary pulse with unit energy and duration $T_s < T$, and $p(t)$ has a duration of $T_P = NT$. Each pulse is modulated with a frequency of $f_n = c(n)/T_c$, where $c(n)$ is a permutation of the integers 1,2,...N. The set $p_k(t) = p(t)e^{j2\pi k f_0 t}$ is orthogonal for $k = 1, 2, ...N$. The spectrum of an OFDM based MC-UWB signal is shown in Figure 4.1 [Web08a].

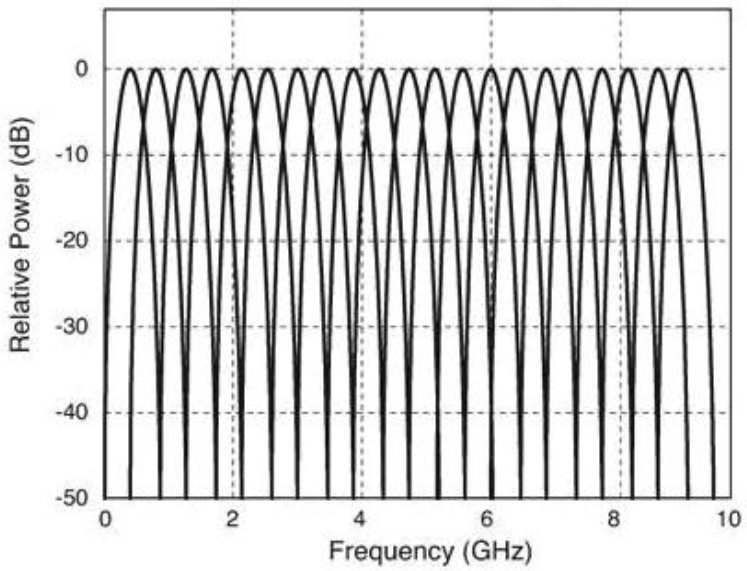

Figure 4.1.: Spectrum of an MC-UWB signal [Web08a].

Unlike the MC-UWB systems, the pulsed UWB systems do not use a modulated sinusoidal carrier to transmit information. The radiated signal is a series of baseband pulses. These pulses are extremely short, commonly in the nanosecond range or shorter. The unmodulated signal as seen by the receiver, in the absence of channel effects, can be represented as

$$s(t) = \sum_{n=-\infty}^{\infty} a_n(t) p(t - nT), \tag{4.3}$$

where $a_n(t)$ is the amplitude of the pulse, $p(t)$ is the received pulse shape and T is the repetition time, defined as the time interval in which one pulse is transmitted. Another important parameter of a UWB transmission is the *pulse repetition frequency* (PRF), defined as $PRF = 1/T$. The duty cycle time, in expression the time in which the pulsed signal is present, is almost always less than 1. Most practical UWB systems make use of some form of pulse-shaping to control the spectral

content, typically a Gaussian pulse shape is implemented. As an example Figure 4.2 shows the spectrum of a UWB pulse train with a spectral pulse with of 4 GHz and a $PRF = 2.5\ MHz$.

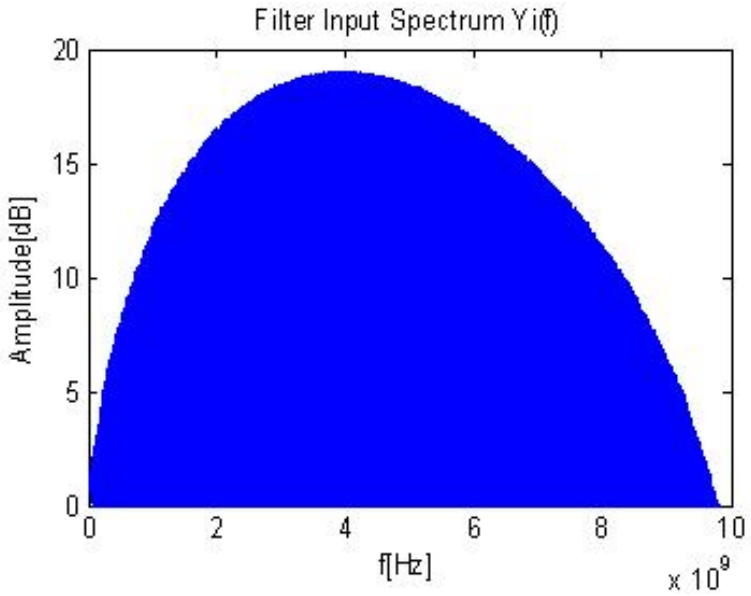

Figure 4.2.: Spectrum of an UWB pulse train [Web08a].

4.1.2. Pulsed Interference

Pulsed interference primary originates from radar. Therefore this subsection focuses on the examination of *Distance Measuring Equipment* (DME) and *TACtical Air Navigation* (TACAN) signals, which are considered to have the most severe effects on GNSS in the L5/E5a band due to their high peak powers.

The DME system is a pulsed ranging system which provides range measurements between an aircraft and a specific ground station. This system is internationally standardized and operates in the $960 - 1215\ MHz$ ARNS frequency band. Its maximum range is about $370 - km$, but at a flight level of $3000 - 6000\ m$ its range is reduced to about 120 km because of line-of-sight propagation [Bas04].

The TACAN system was designed primarily as a military system providing both range and azimuth measurements. The ranging function of this system is identical to that of the DME and the azimuth information is retrieved thanks to a rotating antenna at 900 revolutions per minute [Bas04].

The main purpose of DME/TACAN is distance measurement between an aircraft and a ground station, this measurement works as follows: The aircraft transmits a pulse pair to the ground station which responds with a new pulse pair. The time it takes from receiving the pulse pair to transmitting the new one in the ground station is fixed to a certain value. This enables the interrogating aircraft to calculate the traveled time of the signal by subtracting this fixed delay and dividing by two. Pulse pairs are used because of the greater resistance to background noise in contrast to using only one pulse [Web08a].

A basic pulse pair which constitutes the signal transmitted by each DME/TACAN ground beacon may be assumed as having a Gaussian-shape with a half-amplitude duration of $3.5\mu s$. The pulse pair $s(t)$ can be described by Equation 4.4,

$$s(t) = e^{-\frac{\alpha}{2}t^2} + e^{-\frac{\alpha}{2}(t-\Delta t)^2}, \tag{4.4}$$

where $\alpha = 4.5 \cdot 10^{11}\ s^{-2}$ and the inter-pulse interval is $\Delta t = 12\ \mu s$ [Bas04]. Figure 4.3 illustrates such a pair in the time domain and in the frequency baseband.

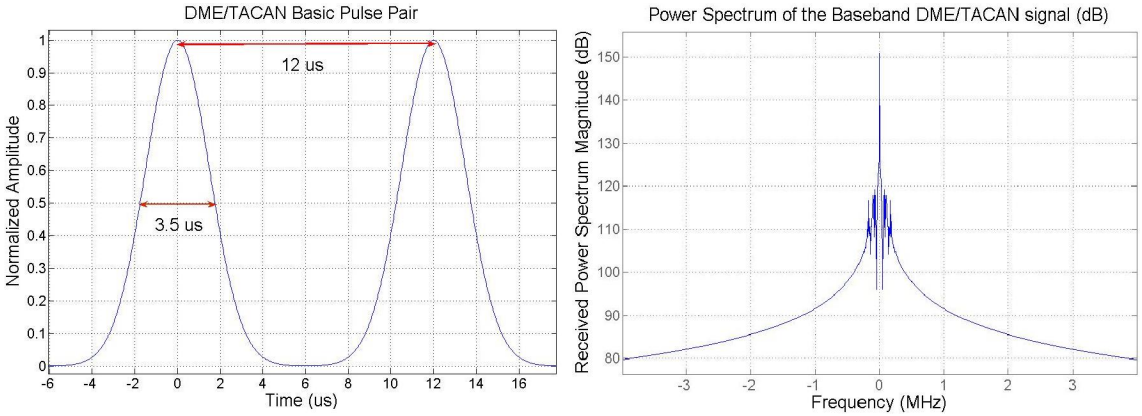

Figure 4.3.: Ideal DME pulse pair in time domain and frequency baseband [Web08b].

The pulse pairs are transmitted on different frequency bands, depending on whether it is transmitted from the aircraft or the ground station. Only the pulses radiated by the ground station, namely the channels 77X to 101X (1164 MHz to 1188 MHz), fall directly into the L5/E5a frequency band and are therefore relevant for the GPS /Galileo in band interference. But due to the 24 MHz passband filter centered at 1176.45 MHz with a not infinitely steep cutoff rate, a certain number of DME channels above 101X and below 77X also interfere with the L5/E5a band. Considering the *pulse pairs per second* (ppps) and *peak pulse power* (pps), the aircraft interrogator has a pps which is in the order of 50 KW and 2 KW, and sends about 5 and 150 ppps, depending whether it has already acquired the DME signal. The ground stations have a pps range of 1 KW to 20 KW and a ppps between 700 and

2700, depending on whether the receiver is close or far away to a hot spot (a location where the impact of DME/TACAN on the victim GNSS receiver is the largest). Military TACAN sites transmit with an often higher pps and an additional 900 ppps as reference marks for their rotating antennas with a maximum rate of 3600 ppps [Bas04][Web08a].

4.1.3. Narrowband Interference

Narrowband RFI only corrupts a part of the bandwidth of the victim GNSS frequency band. It can either originate from the harmonics or intermodulation products of various ground and airborne transmitters (such as the harmonics or intermodulation products of TV and radio broadcasts), out-of-band interference caused by nearby transmitters coupled with inadequate RF filtering in the GNSS receiver, or accidental transmission of signals in the wrong frequency band by experiments [BWP96]. Further narrowband RFI can be distinguished to be a pure tone, also called *Continuous Wave* (CW) interference, as it concentrates all of its power at one frequency. When the CW signal is modulated with a AM or FM it becomes a narrowband interference signal, but as long as the amplitude of the CW signal stays fairly constant it can be still interpreted as CW interference.

4.2. Assessment of Interference Effects on the GNSS Receiver

4.2.1. Assessment of WB interference

WB interference mainly affects the ADC as it reaches trough the complete receiver front-end. Therefore the properties of the UWB-inducted signal at the ADC input are determined by the parameters of the UWB signal such as PRF, bandwidth, power, etc and the frequency response of the whole receiver front-end. So the UWB emission causes different effects on the victim GNSS receiver which are depicted as follows:

(a) An increase of the receiver noise floor (i.e. the AGWN interference assumption), which is equivalent to the degradation of C/N_0 of the GNSS signals. The degradation (in dB) of the noise floor is given as

$$D = 10log(10^{N_0/10} + 10^{I_{UWB}/10}) - N_0, \tag{4.5}$$

where N_0 is the power spectrum density of the receiver thermal noise and I_{UWB} is the power spectrum density of the noise-like UWB interference in the victim receiver input, in dBW/Hz respectively. This AWGN-interference assumption is valid for the following cases:

- For pulse-based UWB devices with PRF dithering, UWB devices will still appear as AWGN if $PRF > BW/k$, where BW is the receiver bandwidth, and k corresponds to the spreading factor.
- A sufficient number of non-synchronized UWB interferers disturbs a victim GNSS receiver. This number depends on the UWB signal type, e.g. for smaller PRF of pulse-based UWB devices, the number is larger.
- MC-OFDM UWB signal without frequency hoping.

(b) A different impact on the GNSS receiver occurs, when the UWB transmission has strong spectrum lines within a GNSS-band. This will result in a CW-like interference.

(c) The third kind of impact of a UWB emission is a pulse-like effect. It occurs if the PRF is quite low so that the victim GNSS receiver can distinguish individual UWB pulses [Web08a].

4.2.2. Assessment of narrowband interference

As mentioned above, narrowband RFI has a small bandwidth relative to the GNSS (in case of the GPS C/A-code 2 MHz). The spread spectrum properties of the GNSS signal enable the receiver to mitigate the influence of narrowband interference. So, in order to show the good resistance of the GNSS signal against interference consider a signal satisfying the equation

$$s(t) = \underbrace{\sqrt{P_{C/A}} C_{C/A}(t) N(t) cos(2\pi f_0 t)}_{GPS-Signal} + \underbrace{\sqrt{2P_J} cos(2\pi(f_0 + f_J)t + \theta_J)}_{Interference}, \tag{4.6}$$

where the first part represents the GPS baseband signal and the second part the interference. At the input of the receiver the *signal-to-interference ratio* (SIR) is simply P/P_J. For GPS the received interfering power can be many times greater than the desired satellite power, because after the correlation process the prior spread GPS signal is concentrated in a narrowband and the prior narrowband RFI is now spread, so that only P_J/PG of the interference power appears in the de-spread

GPS signal bandwidth. PG is called the processing gain and is defined as

$$PG = \frac{B_{code}}{B_{data}}, \tag{4.7}$$

where B_{code} is the bandwidth of the spreading code and B_{data} is the bandwidth of the transmitted data. In order to illustrate the effect of the correlation process, Figure 4.4 shows the signal before and after the correlation.

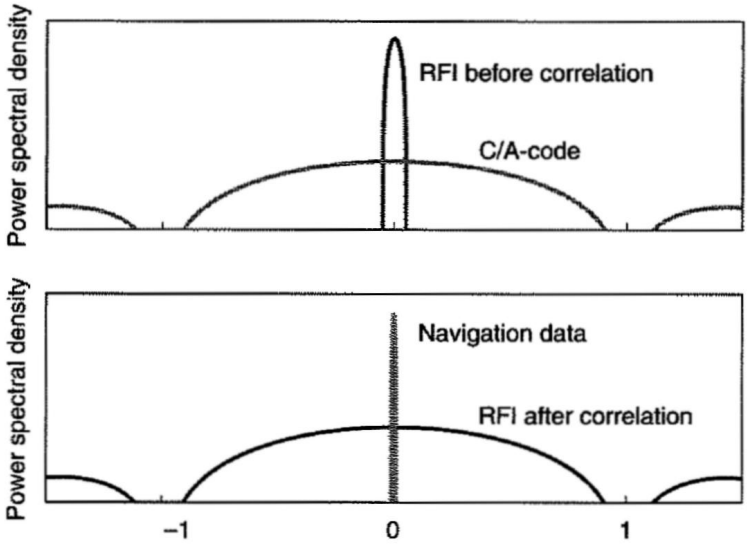

Figure 4.4.: Top: Signals before the correlation; Bottom: Signals after the correlation [Eng06].

But there have to be taken more effects of narrowband interference into consideration, for which it is necessary to examine the spectral characteristic of the GPS C/A-code one more time. As shown in Section 2.4, the GPS C/A-code is a Gold code with a short 1 $msec$ period; which means that the PRN sequence repeats every millisecond. Because of this, the C/A-code does not have a continuous power spectrum. Instead, it has a line spectrum that is spaced by the inverse of the code period, which is 1 KHz apart (see Figure 2.9). As a result, a CW interference can mix with a strong C/A-code line, leak through the correlator and degrade the C/N_0.

So CW interference will have severe effects on the GPS C/A-code signal when it hits one or more of these vulnerable lines. In order to clarify this, Figure 4.5 shows as an example, the C/N_0 of a received L1 signal from a satellite with a specific power and Doppler frequency changing from 0 KHz to 10 KHz, and a CW interference with a specific power at 14 KHz away from the band center at L1 frequency. The

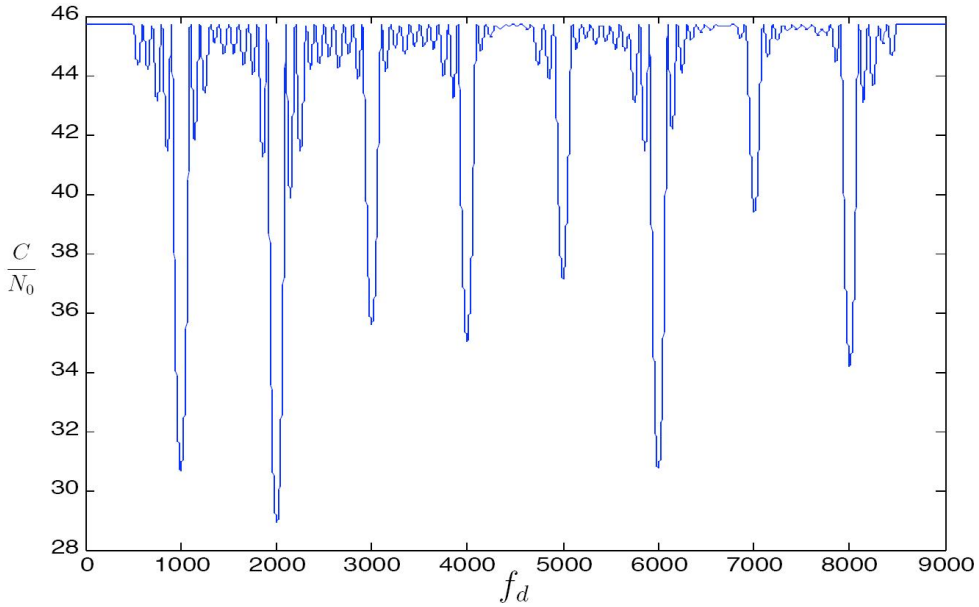

Figure 4.5.: C/N_0 degradation depending on the Doppler frequency [Bal07].

deep troughs in this graph correspond to the coincidence of CW RFI with the code spectral lines. They have different values for the C/N_0, because the lines of the C/A-code have different coefficients in the code spectrum. This strong degradation of the C/N_0 can cause the GPS signal to be lost, depending on the received signal power, interference power, noise power and if a C/A-code line is hit by the CW interference [Bal07]. The P(Y)-code does not have this vulnerability because its code period is one week long.

Due to the fact that there no analog circuits or devices capable of performing the analog correlation for long spreading sequences exist, the signal needs to be digitized first to perform a digital correlation.

In order to digitize the signal a ADC with a particular quantization range is used. The GNSS signal power is adjusted to this range for the interference free case. But when interference, such as CW from the harmonics of AM/FM broadcasts occurs, a serious vulnerability for the pre-correlation A/D-conversion exists. The former statistics of the zero crossing of the incoming signal are no longer determined by a combination of random noise and the GNSS signal (which has a Gaussian shape), but become dominated by the CW signal. The probability density function of a CW signal, i.e. a sine, is given by

$$P(x) = \frac{1}{\pi\sqrt{1-x^2}}. \tag{4.8}$$

This function is plotted in Figure 4.6.

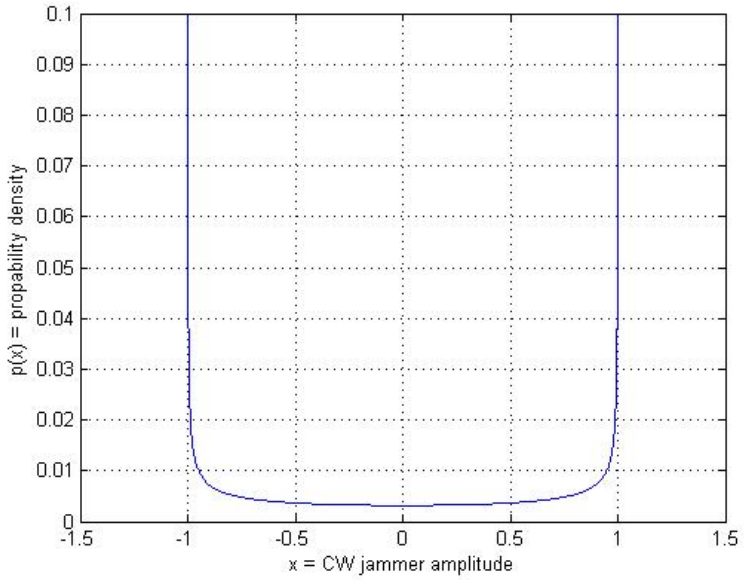

Figure 4.6.: Probability density of a CW (sinusoidal) signal.

As one can see, the statistics of a CW signal imply that it spends most of its time near the peak amplitudes rater than in the vicinity of the zero crossing. The combination of signal plus noise plus CW signal will show the same properties as the CW signal alone, as it dominates the ADC. Hence, there is very little correlation possible in presence of CW interference that captures the pre-correlation A/D converter. In order to avoid this problem one can, as mentioned above, perform the correlation in the analog domain, which is not state of the art, or adjust the quantization thresholds which is described in Section 3.3.

4.2.3. Assessment of Pulsed Interference

Due to the fact that pulsed signals, such as DME, have high peak powers, they can seriously affect the receiver's electronics.

In the receiver's front-end, pulsed interference may cause saturation effects in the LNA and/or might even damage the front-end electronics. Hence a GNSS receiver should make use of an RF power limiter to protect the hardware components from pulsed interference. Also, high power pulse interference is expected to cause saturation to the ADC and so to affect the correlator loops.

Pulsed interference can be categorized by the pulse duration and duty cycle. The

pulse duration is the time width of an individual pulse and the duty cycle is the percentage of time that the interfering pulses are on. If the pulse duration is small compared to the GNSS data bit duration and the duty cycle is less than 10%, pulsed interference is in general no problem for the signal acquistion and tracking even if it is very powerful [Eng06]. But because of the fact that during the pulse time no signal evaluation can take place, pulsed RFI will lead to an increase of the received noise floor.

5. Interference Detection

There are several ways to detect interference in a GNSS receiver. Detection algorithms can work before and after the correlation process. The interference detection algorithms presented in this chapter all depend on pre-correlation techniques. So they use the digitized samples right after the ADC to perform the interference detection.

5.1. Interference Detection by Monitoring the AGC Gain

There are several ways of RFI detection depending on the digitized samples with their current AGC gain. In Section 3.3.2 it is explained that the AGC gain setting can be performed on the estimated input standard deviation by drawing conclusions from the ADC output power. So energy fluctuations at the ADC output will lead to an adaption of the gain to keep the input power constant. Therefore Figure 5.1 shows the AGC gain change as a function of ISR, which is a function of

$$g = g_0 \cdot erf\sqrt{\frac{P_S}{2P_N}}, \tag{5.1}$$

where g is the AGC gain, g_0 is the gain for the RFI-free input and $\sqrt{P_S/2P_N}$ is the SNR [Spi77].
As one can see an increase of the received noise floor is detected by the AGC and the receiver decreases its internal gain to conserve the optimal input standard deviation. So the detection of potential interference is clearly possible by monitoring gain variations with respect to the nominal case.

However the issue of considering only the AGC gain is that the receiver is not specially designed to detect and cope with interference different form that of Gaussian noise, in other words the AGC gain may only be driven by Gaussian noise. If other kind of interference occurs the adaption will not be optimal and lead to higher quantization losses. Therefore, in Section 6.1 a gain steering strategy is presented which

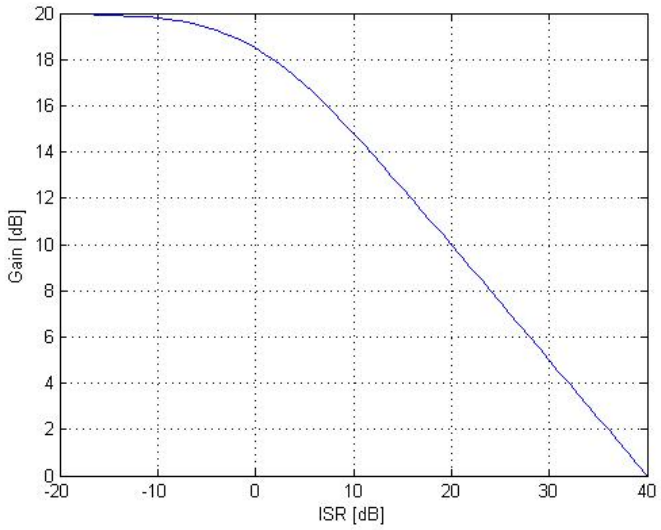

Figure 5.1.: AGC gain change as a function of the input ISR.

makes use of the characteristics of the CW-interference to perform an optimal gain setting. But anyway, in this thesis the quantization loss can be neglected due to the high resolution, i.e. as one can see from Figure 3.2 the correlator-output degradation curves become quite flat for high bit ADCs. The main objective of the AGC here is to avoid saturation effects.

So in order to detect wideband interference with a time constant larger than the pre-detection integration time one may observe the AGC gain. For different kind of interference one may examine the digitized samples obtained at the output of the ADC.

5.2. A Statistical Approach for RFI Detection: The Large Sample T-Test

Due to the fact that the GNSS signal is below the noise floor and the properties of the RFI are unknown, RFI detection can be performed by monitoring statistical changes of the digitized samples. In this context the "Large Sample T-Test" is introduced which is presented here in short, for a detailed description see [Mar04].

The large sample T-Test implies the following assumptions which are valid for the received signal:

- The samples are sufficiently independent.
- RFI will persist for an unknown time span and is not just a single sample time event.
- The distribution functions of the estimated signal parameters are unknown.
- The distributions of the undistorted and RFI distorted signal may not be identical, including the variances between the two population groups.
- An RFI-free signal period is available to perform a statistical assessment of the process.
- A *Wide Sense Stationary* (WSS) time process, i.e. no dynamics are present in the region of investigation.

The algorithm presented in [Mar04] incorporates an assessment window consisting of n samples, which allow to determine the statistical properties of the random process. In the time period of gathering the statistical properties of the process, it is assumed that no RFI is present in the signal. The larger n is selected, the more accurate the assessment of the statistical properties of the process, i.e. the more accurate the calibration. The upper limit for n is determined by the time span in which the process can be considered to be WSS. In addition, the proposed method also incorporates an evaluation window of size k samples, which is shifted over the incoming data stream as shown in Figure 5.2.

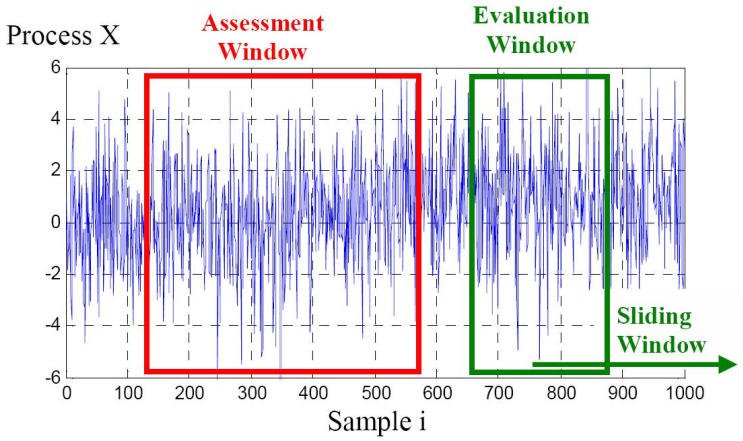

Figure 5.2.: Assessment and evaluation window of large sample T-Test [Mar04].

The estimated value of the evaluation window is tested on a possible inconsistency against the assessment window through a statistical hypothesis test. The following

sections, give an insight into how RFI detection algorithms work on the basis of signal statistics obtained in the assessment window. These hypothesis tests are performed in the time- and frequency domain as well as on the statistical properties of the incoming signal.

5.3. Interference Detection by Monitoring Time Domain Energy Fluctuations of the Digitized Samples

RFI can be detected by monitoring power fluctuations of the received digitized samples, because in general, any interference source increases the received signal power. The signal power can be computed with the following Equation

$$U = E\left[X^2\right] = \frac{1}{N}\sum_{i=1}^{N} X_{Qi}^2, \tag{5.2}$$

where X_{Qi} is the received digitized sample at time i. Power fluctuations can be detected by means of the large sample T-Test algorithm presented in 5.2 and [Mar04]. Assume that for $t < T_{RFI}$ no interference is present, the received signal can be described as $X_{t<T_{RFI}} = s(t) + n(t)$. For $t \geq TRFI$ the received signal is defined as $X_{t\geq T_{RFI}} = s(t) + n(t) + j(t)$, where $s(t)$ is the GNSS signal, $n(t)$ is noise and $j(t)$ is RFI.

Because for $t < T_{RFI}$ the received signal signifies a noise sequence therefore the signal can be characterized as a WSS process, so it is valid to collect the needed signal characteristics within this timespan.

The assessment window of the energy fluctuation detector is introduced in Equation 5.3

$$U_{AssessmentWindow} = \frac{1}{K}\sum_{i=0}^{K-1} X_{Qi}^2, \tag{5.3}$$

where K is the assessment window length in samples.

The evaluation window is defined accordingly in Equation 5.4

$$U_{EvaluationWindow} = \frac{1}{N}\sum_{i=0}^{N-1} X_{Qi}^2, \tag{5.4}$$

where N is the evaluation window length in samples.

So if the estimated power within the assessment window differs from that in the evaluation window one can suggest that interference has occurred.

This algorithm of detecting temporal energy fluctuation is especially important for the detection of pulsed-RFI in the time domain. According to this in [Gra02] a digital pulse blanking technique is proposed, where an ADC having more bits than required in the digital tracking loops is used. In this published work, 8 bits were used to quantize the signal but the signal was eventually represented using only 3 of those 8 bits for further digital processing. The additional available bits were used to implement digital blanking, as explained in Chapter 6. On the left, Figure 5.3 shows the noise signal and the additional pulsed RFI which reaches in the dynamic range, in the time and on the right one can see the typical ADC bins loading when only noise is present as one can see, the detection threshold is at digital 8. The sense

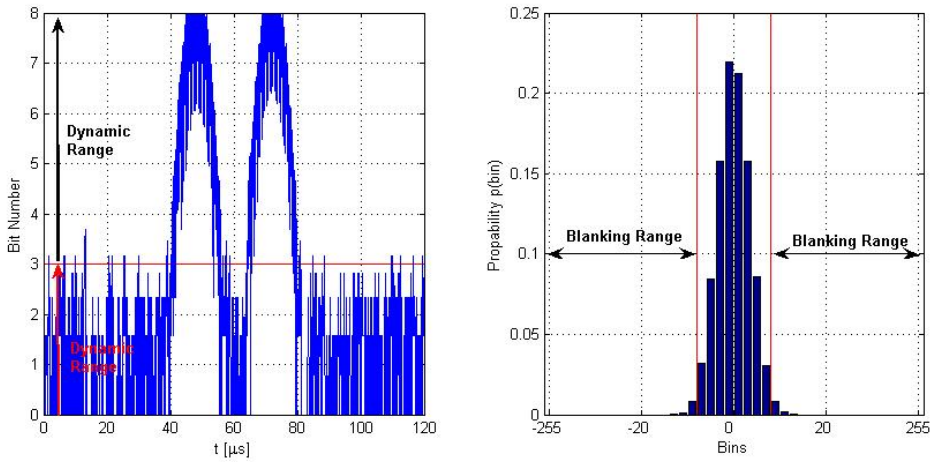

Figure 5.3.: Dectection of pulsed interference, left time domain plot for noise and pulsed RFI; right possible 8-bit bins loading in presence of thermal Gaussian noise only and the temporal detection thresholds, respectively.

of using a dynamic range is presented in Chapter 6 when it comes to the mitigation of pulsed-RFI. At this point one needs only to identify the samples belonging to the pulsed RFI, so that these samples exceeding that threshold, might not be used to drive the AGC. Because if those temporal high power fluctuations would be respected, the gain would be set to low.

This pulse detection strategy proposed in [Gra02] is well known not to be the best, but it is much simpler than others, because no pulse detector circuit is required to identify the beginning and end of each pulse. Further, the implementation does not need memory to track samples that are part of a pulse.

A different, but more complicated strategy to detect temporal power fluctuations is to make use of a moving average filter. The structure of it is presented in Figure 5.4.

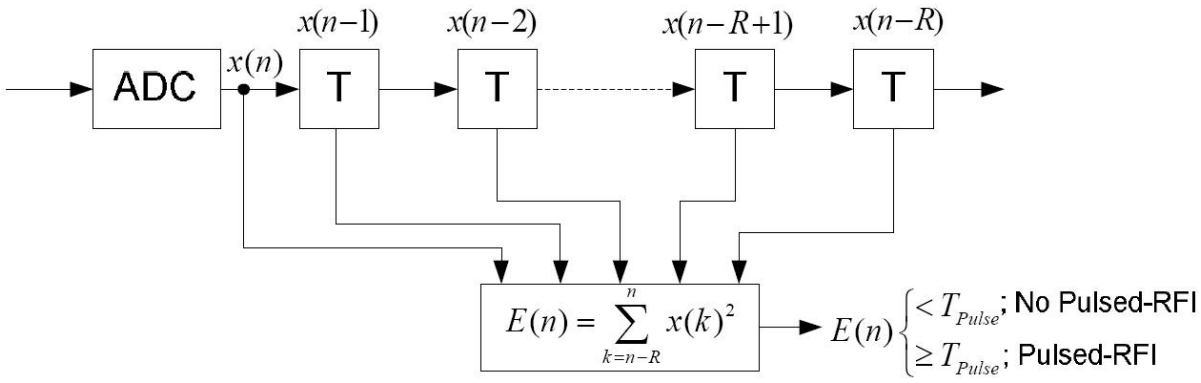

Figure 5.4.: Shift register for the detection of power fluctuations.

At each clock the quantized samples are shifted one register further and the total power in the whole shift-register is estimated. If it exceeds a certain threshold, pulsed RFI is declared to be detected. The detection threshold can be defined in addiction to a certain false alarm rate, but therefore it is necessary to determine the PDF of the output $E(n)$. The PDF of $E(n)$ with a total shift register length $N_{Reg} = 20$ is plotted in Figure 5.5 for the case that only white Gaussian noise was present at the input of the ADC. Here the ADC had 3 *bit* to represent the signal and 5 *bit* dynamic range, like the one in [Gra02].

As one can see, the values of $E(n)$ follow a Chi-Square distribution, with $N_{Reg} - 1$ degrees of freedom.

$$PDF(E) \sim \chi^2_{N_{Reg}-1} \tag{5.5}$$

In this thesis the pulse detection value T_{Pulse} was defined similar to the method proposed in [Gra02], relying on the fact that the pulses are short and have a great energy compared to the noise floor. So when total register length is N_{Reg} the digital threshold is computed as follows

$$T_{Pulse} = N_{Reg} \cdot (2^{bit_{res}} - 1)^2, \tag{5.6}$$

where bit_{useful} is the number of initially used bits to represent the signal. In this work a total register length of $N_{Reg} = 20$ was taken, notice, N_{Reg} should not be longer as the number samples representing a single pulse, else this threshold definition might lead to a not detection of the pulses, and one should choose it in addiction to a certain pre-defined false alarm rate.

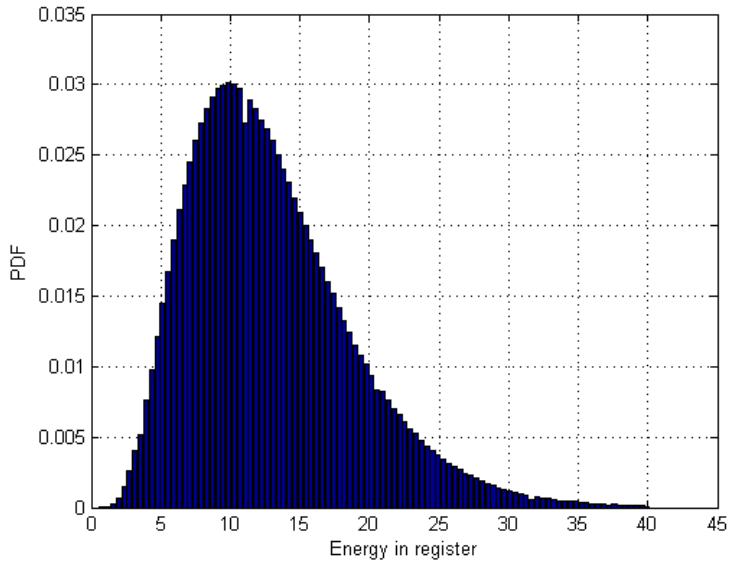

Figure 5.5.: PDF of Energy in shift register.

The structure presented in Figure 5.5 has the benefit that in contrast to the method proposed in [Gra02] the algorithm can better differentiate between noise, CW and pulsed RFI, because as one can see in Figure 5.6, the power estimator works as a low pass filter, that estimates the "envelope" of the pulses, notice that "envelope", i.e. the computed power in the register was weighted here with a factor 1/100 for illustration purpose.

Also as one can see in this figure, this approach of pulsed interference detection enables to estimate the start and end time of a pulse so that even those samples instantaneous before and after the pulse can be excluded, as they are also corrupted by interference.

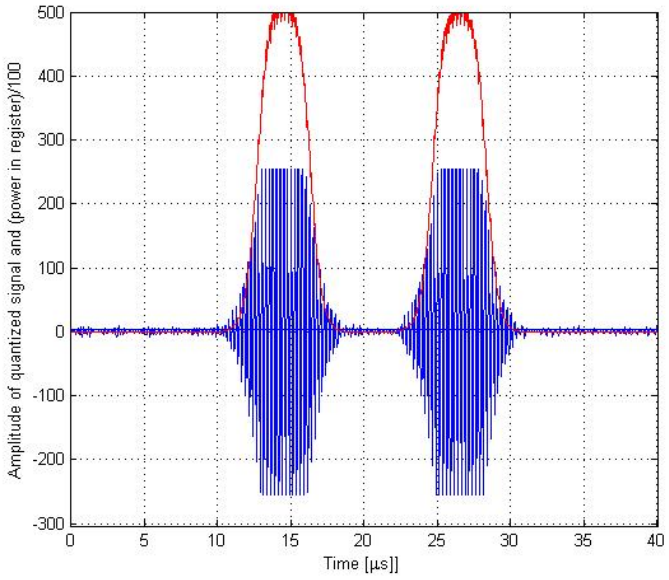

Figure 5.6.: Detection of power fluctuations using a shift register, blue: the incoming signal disturbed by DME pulses: red: the estimated power in the shift register $\cdot 1/100$.

5.4. Chi-Square Test at the ADC Level

As explained in Section 3.2, the ADC used in this work has a high resolution so that the distribution of the incoming signal will be clearly represented on the ADC bins. Recall that, in nominal conditions, the ADC bins distribution is Gaussian, because the receiver thermal noise is prevalent at this level, and is maintained constant as a result of the AGC gain adaptation. If an interference source is introduced, AGC gain decreases to conserve the Gaussian shape. In former ADCs, the ADC distribution may seem nominal even in presence of interference due to the little ADC bins. However if the resolution is high enough, ADC distribution may clearly represent the distribution of the interfering signal. Indeed, the incoming signal distribution shape is unchanged by the AGC that only applies a gain [Bas04].

When CW like interference is present, the PDF of the signal at the ADC input is changed. In this case, its PDF can be defined as follows [Kon08]

$$p(x) = \frac{1}{\pi\sqrt{2\pi\sigma_n^2}} \int_0^\pi exp\left(-\frac{(x - Acos\theta)^2}{2\sigma_n^2}\right) d\theta, \tag{5.7}$$

where σ_n is the variance of the receiver thermal noise at the ADC input and $A^2/2\sigma_n^2$ is the CW interference-to-noise ratio (INR). Several curves for p(x) obtained by numerically evaluating Equation 5.7 are shown in Figure 5.7.

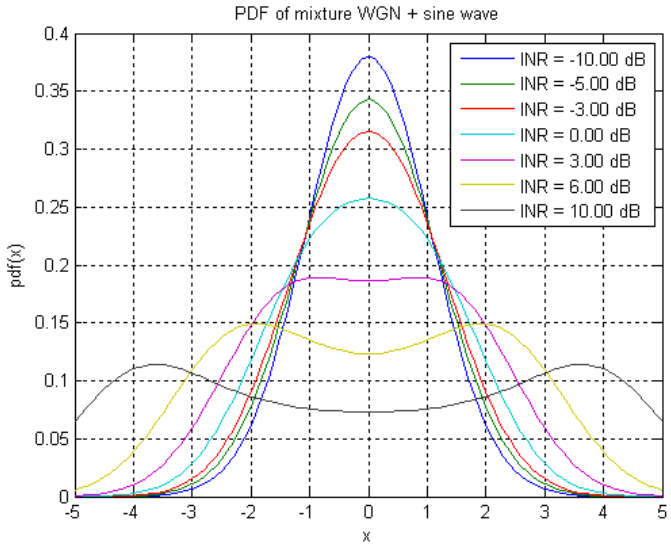

Figure 5.7.: Probability density function of CW plus noise [Kon08].

Thus it is possible to implement a test on ADC bins distribution to detect interference. A straightforward approach is to use the Chi-Square test which may be used

to decide whether two sets of data are consistent. The nominal distribution may be stored during an initialization phase when the receiver is turned on, assuming of course that no interference is present or, as done in this thesis, one can use the Gaussian CDF and compute the theoretical bins loading as follows

$$n_{nom}(i) = normcdf(P(i)/\sigma) - normcdf(P(i+1)/\sigma), \tag{5.8}$$

where P is the partition of the quantization intervals as defined in Equation 3.4 and σ is the standard deviation of Gaussian noise [Kon08].

When the standard deviation of noise σ is set according to the optimum ratio $k_{opt} = L/\sigma$, the equation above can be modified. Normalizing $L = 1$ one can compute the optimal bin loading as follows

$$n_{nom}(i) = \frac{1}{2} erfc\left(\frac{P(i)k_{opt}}{\sqrt{2}}\right) - \frac{1}{2} erfc\left(\frac{P(i+1)k_{opt}}{\sqrt{2}}\right). \tag{5.9}$$

In this work the Chi-Square test was not performed on an ADC with a dynamic range. When the optimal bin loading shall be computed for an ADC with a dynamic range Equation 5.9 needs to be modified so that $L_{new} = 1$, see Equation 3.7.

After the optimal bin loading is estimated, the Chi-Square Test can be performed. It is defined as

$$BinT = \sum_i \frac{(n_{current}(i) - n_{nom}(i))^2}{n_{current}(i) + n_{nom}(i)}. \tag{5.10}$$

The sum is defined over all bins. A large value of $BinT$ indicates that the null hypothesis, which is n_{nom} and $n_{current}$ terms are drawn from the same distribution, is rather unlikely. If the number of bins is large, the chi-square probability function is a good approximation to the distribution of $BinT$ in the case of the null hypothesis. If data are collected in such a way that the sum of n_{nom} equates the sum of $n_{current}$ then the degrees of freedom of the Chi-Square distribution is equal to the number of bins minus one. As an example, Figure 5.8 shows the theoretical PDF and the experimental PDF of the Chi-Square distribution with 5 degrees of freedom for a 2.5 bit ADC.

Having knowledge about the PDF of $BinT$, one can define the threshold $T_{ChiSquare}$ according to a certain false alarm probability. In this work that threshold was estimated by means of the Monte-Carlo runs, because the parameters of the theoretical PDF could not be found. The problem to determine the theoretical PDF for $BinT$ is also depicted in the work [Kon08], where the theoretical PDFs were always overbound to the ones obtained by Monte Carlo runs. Interference detection can be performed on the basis of the Neumann-Person Test, with which one can decide if

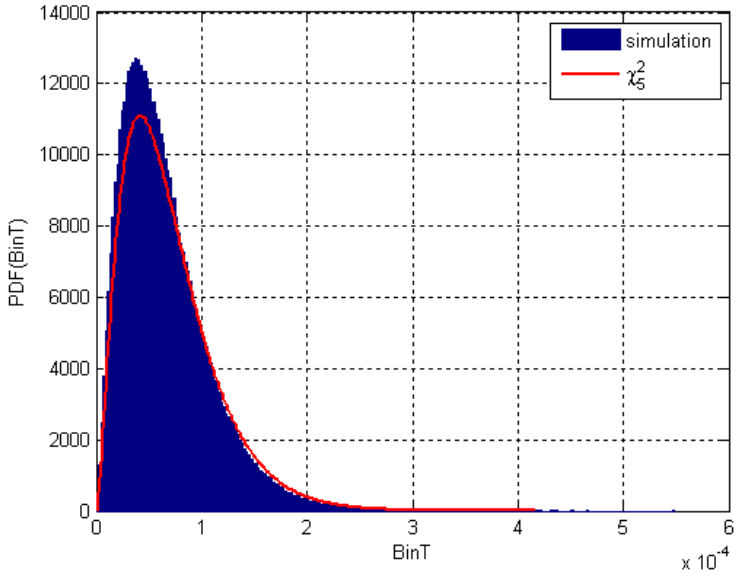

Figure 5.8.: Theoretical and experimental PDFs of $BinT$ test statistic in interference free case for a 2.5 bit ADC [Kon08].

to sets of data are collected from the same distribution

$$BinT \begin{array}{c} H_0 \\ < \\ > \\ H_1 \end{array} T_{ChiSquare}. \tag{5.11}$$

In order to clarify how it works, Figure 5.9 shows the theoretical and experimental obtained Chi-Square PDF of noise superposed with a CW interferer with an INR of 10 dB. Focusing on the x-axis one sees that the values of $BinT$ are much larger as for the CW free case. So a interference detection monitoring the ADC bins is clearly possible. For a description of the histogram based interference detection in more detail see [Kon08].

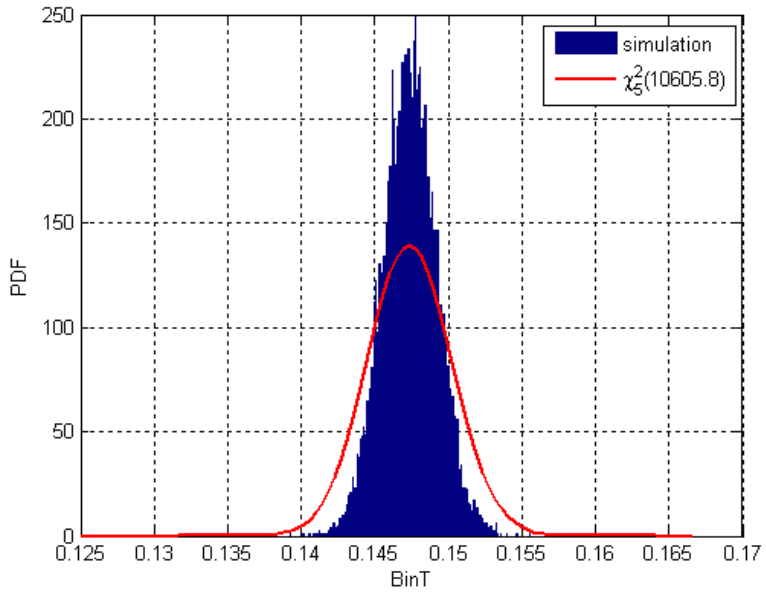

Figure 5.9.: Theoretical and experimental PDF of $BinT$ test statistic for a CW-interference with 10 dB INR, obtained for a 2.5 bit ADC [Kon08].

5.5. Interference Detection by Observing the Power Spectral Density

The methods of RFI detection treated up to this point, all worked in the time domain. But especially for the detection of narrowband interference it is obvious to transform the received signal into the frequency domain. This transformation is performed by means of the digitized samples with the (FFT). The use of the FFT is possible here, because in contrast to former GNSS receivers this one simulated in this thesis owns a high resolution ADC, so that the estimate discrete frequency spectrum approximates the real continuous one quite close. After that the *Power Spectral Density* (PSD) is examined.

The interference detection algorithm is shown in Figure 5.10.

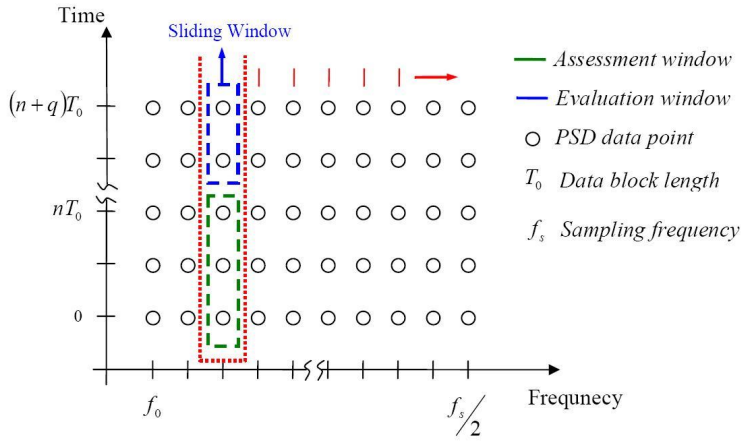

Figure 5.10.: Time-frequency search for power spectral density fluctuations [Mar04].

In the initialization phase, where there is no RFI present ($nT_0 < T_{RFI}$), the power spectral density is determined in the assessment window. Due to that fact that during that time interval only white Gaussian noise was present, the spectral density should be constant. After that, as depicted in Figure 5.10, for every block at time instance $0..(n+q)T_0$ the discrete PSD is evaluated as

$$S_x(f) = \mathscr{F}\{x(t)\} \cdot \mathscr{F}\{x(t)\}^*. \tag{5.12}$$

The test-statistic is established over time for every frequency component to the PSD as indicated in Figure 5.10 and in comparison of Equation 5.13 to 5.14

$$E_{AssessmentWindow,f_j} = \frac{1}{n} \sum_{i=0}^{n-1} PSD_{[0,nT_0]}(f_j) \tag{5.13}$$

$$E_{EvaluationWindow,f_j} = \frac{1}{k}\sum_{i=0}^{k-1} PSD_{[mT_0,(m+k)T_0]}(f_j), \tag{5.14}$$

where T_0 is the data block length, n the length of the assessment window, f_j the frequency component to be investigated, m the first sample of the evaluation window, and k the length of the evaluation window. If the results obtained from Equation 5.13 and 5.14 strongly differ, one is likely to suggest that for this frequency component f_j interference has occurred.

The above described approach can be simplified by setting a certain threshold, which the PSD and so the magnitude spectrum of $\mathscr{F}\{X(t)\}$ may not be allowed to exceed. A typical strategy to set such a spectral threshold is to define it according to a certain false alarm rate p_{fa}. But however, in order to do so one needs to have knowledge about the *probability density function* (PDF) of the magnitude spectrum, where the interference detection is performed in this thesis, first.

Consider, in a GNSS system noise is the prevalent signal. The PDF of noise is known to follow a Normal distribution, given with the Equation

$$f_x(x) = \frac{1}{\sigma\sqrt{(2\pi)}} exp\left(\frac{-x^2}{2\sigma^2}\right). \tag{5.15}$$

The continuous Fourier spectrum of noise is well known to theoretically reach from $-\infty$ to $+\infty$, owning a Normal distribution of the amplitude of each frequency component. But however, using a front-end filter limits the signal's spectral range. The discrete spectrum is computed with the FFT using a rectangular window on the basis of blocks with a certain number of samples. Due to the fact that noise has whether odd nor even temporal characteristics, the results of the FFT will be complex, with both, the real and imaginary part having a Normal distribution of the amplitudes of the different frequencies. Figure 5.11 shows the real part of the spectrum of filtered noise and its theoretical PDF in the filter bandwidth, its imaginary part looks quite the same.

As explained above, the result of the FFT of such a block is complex, so for interference detection one would need to search the real and imaginary part of the spectrum for a threshold violation. But this is not the common way in a receiver, the more practical way of interference detection takes place on the basis of the magnitude spectrum, which leads to a different PDF due to the fact that the magnitude spectrum is computed with

$$|X(f)| = \sqrt{\mathscr{F}\{x(t)\} \cdot \mathscr{F}\{x(t)\}^*} = \sqrt{(Re\{X(f)\})^2 + (Im\{X(f)\})^2)}. \tag{5.16}$$

The PDF of the magnitude spectrum of noise follows a Rayleigh distribution, because both the real and the imaginary part are two statistical independent processes,

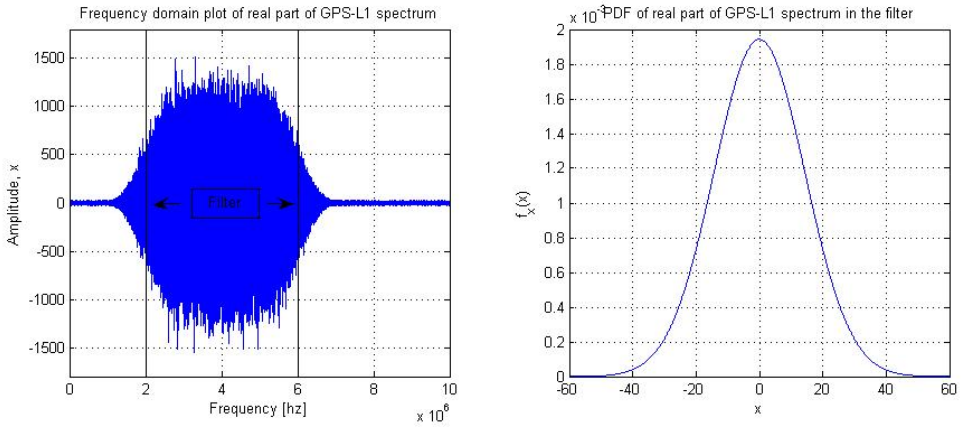

Figure 5.11.: Real part of spectrum of filtered noise and its theoretical PDF in the filter bandwidth.

$\mathscr{R} \sim \sqrt{(Re\{X(f)\})^2 + (Im\{X(f)\})^2)}$. The PDF of the Rayleigh distribution is given by Equation 5.17.

$$f_x(x) = \frac{x^2}{\sigma^2} exp\left(\frac{-x^2}{2\sigma^2}\right). \tag{5.17}$$

Figure 5.12 shows the magnitude spectrum of filtered noise and its theoretical PDF in the filter bandwidth.

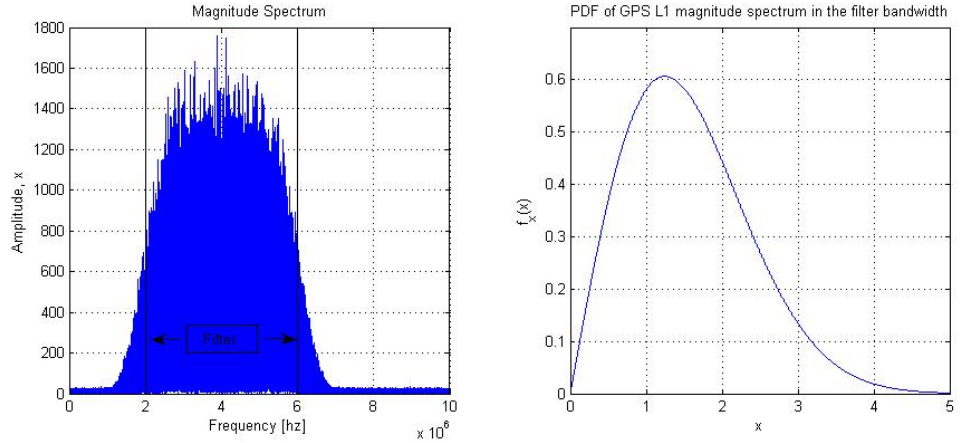

Figure 5.12.: Magnitude spectrum of filtered noise and its theoretical PDF in the filter bandwidth.

So finally one can the compute the theoretical interference detection threshold with the *cumulative distribution function* (CDF) of the Rayleigh distribution and a predefined false alarm rate. But in order to determine the absolute threshold one more parameter needs to be known, i.e. the mean ($\mu_{MS_{AssessmentWindow}}$) or the variance

($\sigma_{MS_{AssessmentWindow}}$) of the magnitude spectrum in the filter bandwidth when no RFI is present. In this thesis the mean of the magnitude spectrum was determined during the initialization time, because MATLAB was not able to determine the correct variance with the "std"-function. For the Rayleigh distribution the mean μ is connected with its variance σ as follows

$$\mu = \sigma\sqrt{\frac{\pi}{2}}. \tag{5.18}$$

The CDF of the Rayleigh distribution suffices the Equation

$$p(x) = 1 - exp\left(\frac{-x^2}{2\sigma^2}\right). \tag{5.19}$$

So with the pre-defined false-alarm-rate p_{fa} and the Equations 5.18 and 5.19, the absolute threshold can be computed with Equation 5.20 according to

$$T_{freq}(p_{fa}) = \mu_{MS_{AssessmentWindow}} \cdot \sqrt{\frac{2}{\pi}} \cdot \sqrt{2 \cdot ln(p_{fa})}. \tag{5.20}$$

Figure 5.13 shows as an example, the Rayleigh CDF for $\sigma = 1$ and according to that the threshold $T_{freq} \sim \mu_{MS_{AssessmentWindow}}$ in the magnitude spectrum for a false-alarm-rate of $p_{fa} = 1$ %.

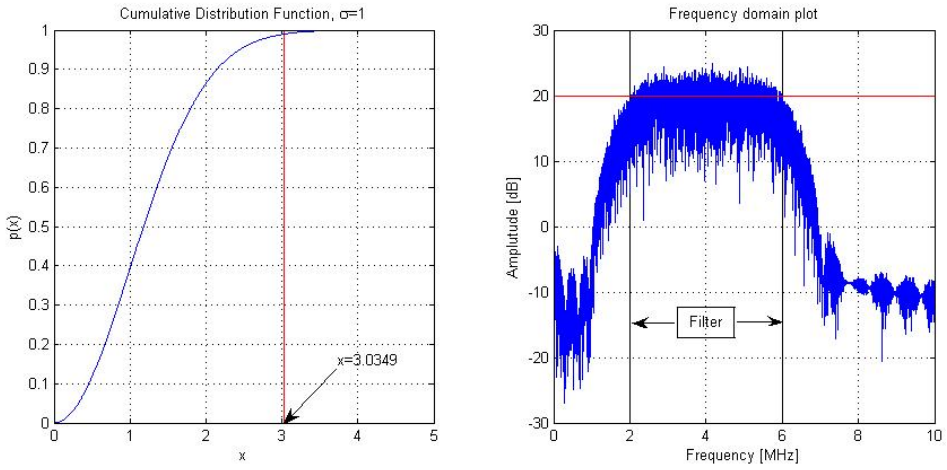

Figure 5.13.: Rayleigh CDF for $\sigma = 1$, $p_{fa} = 1$ % and magintude spectrum with absolute spectral threshold.

Note: As shown in the simulation chapter, the pre-defined false-alarm-rate differs from that in the measurements. This error is a result of the mean computation, because the magnitude spectrum within the filter is assumed to be flat, but due to the filter operation the power in the filter edges is lower than in the middle. But however, this error is, as will be shown, quite small so that the approximation of a rectangular bandpass is valid.

5.5.1. Problems of correct Spectrum Estimation Using the FFT

In the discussion above a spectral threshold was determined on the basis of the Rayleigh CDF, a certain false-alarm-rate and the mean of the magnitude spectrum in the filter bandwidth for no interference present. All FFT points detected above this threshold were assumed to originate form that false-alarm-rate and RFI. But when strong CW-RFI occurs, this approach will lead to a much greater false-alarm-rate as that defined for the spectral threshold. In order to explain this issue, this section examines the "leakage" phenomenon and give an insight into techniques for correct CW-frequency detection even if the interferer power is quite large.

First of all, in order to understand the leakage phenomenon, it is necessary to make a short discussion of the *Discrete-Fourier-Transformation* (DFT).

The DFT, which is the basis of every digital signal analysis, is used to transform a sample sequence into the frequency domain, so, i.e. the DFT gives the same description of a digital system, like the Fourier-Transformation does for a analog one. Its mathematical expression is given by Equation 5.21.

$$X(m) = \sum_{n=0}^{N-1} x(n) exp\left(-j\frac{2\pi mn}{N}\right); \quad m = 0, 1, ..., N-1. \tag{5.21}$$

Here, $x(n)$ is a discrete time chain with N samples and $X(m)$ is the discrete frequency spectrum with its spectral lines at $f_m = mf_S/N$, f_S represents the sampling frequency. The companion to the DFT is the *Inverse Discrete Fourier Transform* (IDFT), which transform the discrete spectral lines $X(m)$, obtained from the DFT of a sample sequence, back into the time domain. Its mathematical expression is given with Equation 5.22.

$$x(n) = \frac{1}{N}\sum_{m=0}^{N-1} X(m) exp\left(+j\frac{2\pi mn}{N}\right); \quad n = 0, 1, ..., N-1. \tag{5.22}$$

But note, in most common digital systems, which need to transform a signal into the frequency domain, do not make use of the DFT or IDFT, they make use of the FFT or IFFT, respectively. Using the FFT or IFFT leads to the same results as the DFT or IDFT, but however, its computation techniques are quite different. In shortcut, in contrast to the DFT, the FFT makes use of a "Butterfly" like structure of the algorithm, with which the FFT can reduce the amount of calculation by limiting the number of points (N in the above formula). The reduction ratio is approximately N^2 versus $2N\log_2(N)$. The best choice for the number of FFT points is to set it to

$N = 2^k,\ k \in \mathbb{N}$. For example if k is 10 ($N = 1024$), the ration is about 1/100. For a description of the FFT algorithm in detail see [Opp99].

Now, that the properties of the DFT and IDFT are discussed, it is necessary to focus on the connection between the DFT and the continuous Fourier transformation. In this context it is most important to examine under which conditions their results are equal.

Therefore consider a continuous time function $s(t)$, which is transformed into its continuous Fourier spectrum using the continuous Fourier transformation

$$x(t) \bullet\!\!-\!\!\circ\, X(f). \tag{5.23}$$

When now both, temporal limitation

$$x(t) = 0 \ for \begin{cases} t < 0 \\ t \geq \frac{N}{f_S} = NT_S = T \end{cases} \tag{5.24}$$

as well as spectral limitation

$$X(f) = 0 \quad for \quad |f| > 2f_S = \frac{1}{2T_S} \tag{5.25}$$

are on the hand, the DFT of the samples $x(nT_S)$ of the continuous function $x(t)$ provides the samples $X(mf_S/N)$ of the continuous Fourier spectrum $X(f)$

$$\{x(nT_S)\} \overset{DFT}{\circ\!\!-\!\!\bullet} \left\{ X\left(\frac{mf_S}{N}\right) \right\}, \tag{5.26}$$

as long as the number N is at least as large that both, $x(t)$ and $X(f)$ are sampled over their whole length.

Backwards, under these conditions, the IDFT of the samples $X(mf_S/N)$ of the continuous Fourier spectrum $X(f)$ provide the samples $x(nT_S)$ of the continuous time function $x(t)$ [Rup00].

Drawing conclusions from the discussion above for the GPS system, the DFT will not provide the correct samples of the GPS spectrum, which is the spectrum noise. For noise, band limitation may easily be obtained using a filter but due to the fact that noise is also infinite in time, and the DFT can only be computed on a finite number of samples, one also needs to limit the signal in time.

The problem that now occurs is that in order to limit a signal in time, one needs to multiply it with a window, commonly its form is of a rectangular shape. But a

multiplication in the time domain is equivalent to a convolution in the frequency spectrum. For this system this means to convolve the frequency samples with a sinc-function. Due to this, a effect called “leakage” occurs, where small amounts of signal energy are observed in frequency components that do not exist in the original signal. This term leakage, refers to the fact that it appears as if some energy has “leaked” out of the original signal spectrum into other frequencies.

However, when it comes to the computation of the RFI free signal spectrum using windows with a length of 1 ms and $f_S = 2 \cdot 10^7$ in this work, the error produced by the FFT will be quite small and can be neglected here.

But when the incoming signal is superposed with a high power RFI signal and the belonging spectrum shall be computed using the FFT, one can no longer neglect the leakage effect. Because now the spectrum may strongly differ from its real one, which serves to a great interference false detection rate using the RFI detection approach presented in Section 5.5. In order to assess the leakage effect, the next subsection gives a detailed examination of the impact of temporal windowing and also techniques to mitigate it by making use of different temporal window types.

Due to the fact that the leakage effect occurred worst for strong CW-RFI powers in this thesis, it is explained on the DFT analysis of sinusoidal signals.

5.5.2. DFT Analysis of Sinusoidal Signals

This section deals with the errors of the spectral estimation when $x(t)$ is periodic, or more special for CW-interference, when $x(t)$ has a sine waveform with infinite duration. For the DFT computation this means that $x(t)$ exceeds the whole sampling interval, with the impact that now the DFT leads to a approximation of the Fourier-chain-coefficients. But also shown here, there are some conditions under which for the sine frequency ω_0, against T_S, the duration of a sampling interval and N the number of samples, leads to the correct spectrum. However, in the real world one cannot choose the CW-frequencies to conform these conditions, so one needs to deal with leakage effects.

For a simple explanation of those spectral estimation errors a periodic signal is chosen to comply with Equation 5.27.

$$x(t) = e^{j\omega_0 t} \tag{5.27}$$

At this place a complex periodic signal is used, because it concentrates all its power at the frequency ω_0 and so it is easier to compute its DFT. The DFT of this signal

is described by the following equation [Ste79]

$$\begin{aligned} X(j\omega) &= \sum_{n=0}^{N-1} s(t) e^{-j\omega n T_S} \\ &= \sum_{n=0}^{N-1} s(nT_S) e^{-j2\omega n T_S} \\ &= \sum_{n=0}^{N-1} e^{j\omega_0 n T_S} e^{-j\omega n T_S} \\ &= \left(1 - e^{N(\omega_0-\omega)T}\right) / \left(1 - e^{(\omega_0-\omega)T_S}\right); \end{aligned} \tag{5.28}$$

and its associated magnitude spectrum is given with

$$|X(j\omega)| = \left| sin\frac{N(\omega_0-\omega)T_S}{2} / sin\frac{(\omega_0-\omega)T_S}{2} \right|. \tag{5.29}$$

Observing this equation it comes up, that when $\omega \to \omega_0$,

$$X(j\omega_0) = N, \tag{5.30}$$

what is the maximum magnitude of $X(j\omega)$. Further one can observe that $X(j\omega)$ is zero when $\omega - \omega_0$ is a multiple of $2\pi/NT_S$, because at these points the whole sampling interval of NT_S seconds obtains a even number of sine cycles. The results of this examination on Equation 5.29 are presented in Figure 5.14, obviously a sine with the frequency ω_0 can misleadingly lead to a DFT at a different frequency, when the N samples reach not about a even number of sine cycles.
Further it is known that the DFT of $x(t)$ would be correct under any conditions if $N \to \infty$. Figure 5.15 shows the estimated spectrum of the signal for different values of N. Obviously, if N is small and the sampling process contains whether a large or even number of cycles of $x(t)$, a large error in the spectrum computation occurs.

5.5.3. Different Window Types for Spectral Leakage Mitigation

In the Figures 5.14 and 5.15 the leakage effect is illustrated. According to this it was shown that those parts of $x(t)$ which are periodic beyond the interval NT_S seem to "leak" into those parts of the spectrum which are close to the correct spectral value. As shown in Figure 5.14 leaking is the direct result of an temporal abscission, i.e. the multiplication with a rectangular window $w(t)$ with the duration NT_S. Especially those sharp corners of $w(t)$ contribute to the leaking in the estimated spectrum.

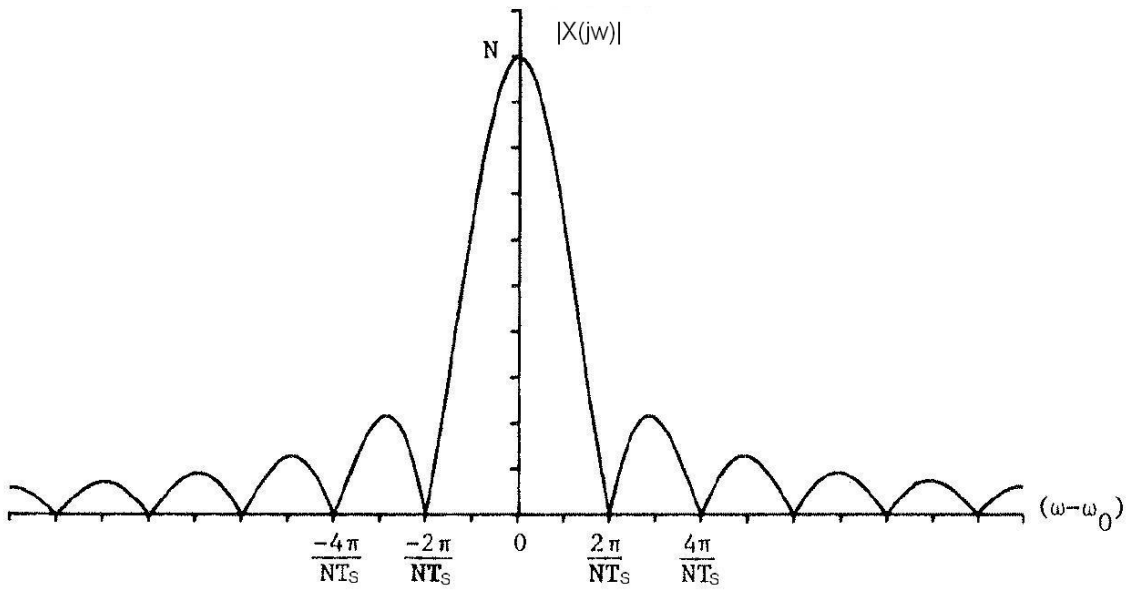

Figure 5.14.: DFT magnitude spectrum computed with N samples a sine with the frequency ω_0 [Ste79].

From the knowledge of that leakage source, one can conclude that the method of leakage mitigation techniques insist in rounding those sharp corners of $w(t)$. So a common approach is to multiply the the signal with a window function, not having such sharp corners as the rectangular window, and than to compute the DFT.

In this context different window types were proposed, for example the Hanning-, Hamming-, Blackman-, Blackman-Harris- or Flat-Top-window. Figure 5.16 shows the spectral characteristic of the different window functions.

As one can see, all window function have a less leakage than the rectangular one on top. In this thesis the Hanning-window was chosen, because it fulfills the requirements best, i.e. it sharply cuts of near its peak, its side lobes fall quite fast and it is easy to build (simple raised cosine). Later when it comes to spectral excision techniques in Chapter 6 this is even more clarified.

5.5.4. The Hanning Window

As explained above, when one wants to compute the FFT over a small time interval (1 ms in this thesis), windowing is necessary to decrease the spectral leakage effect.

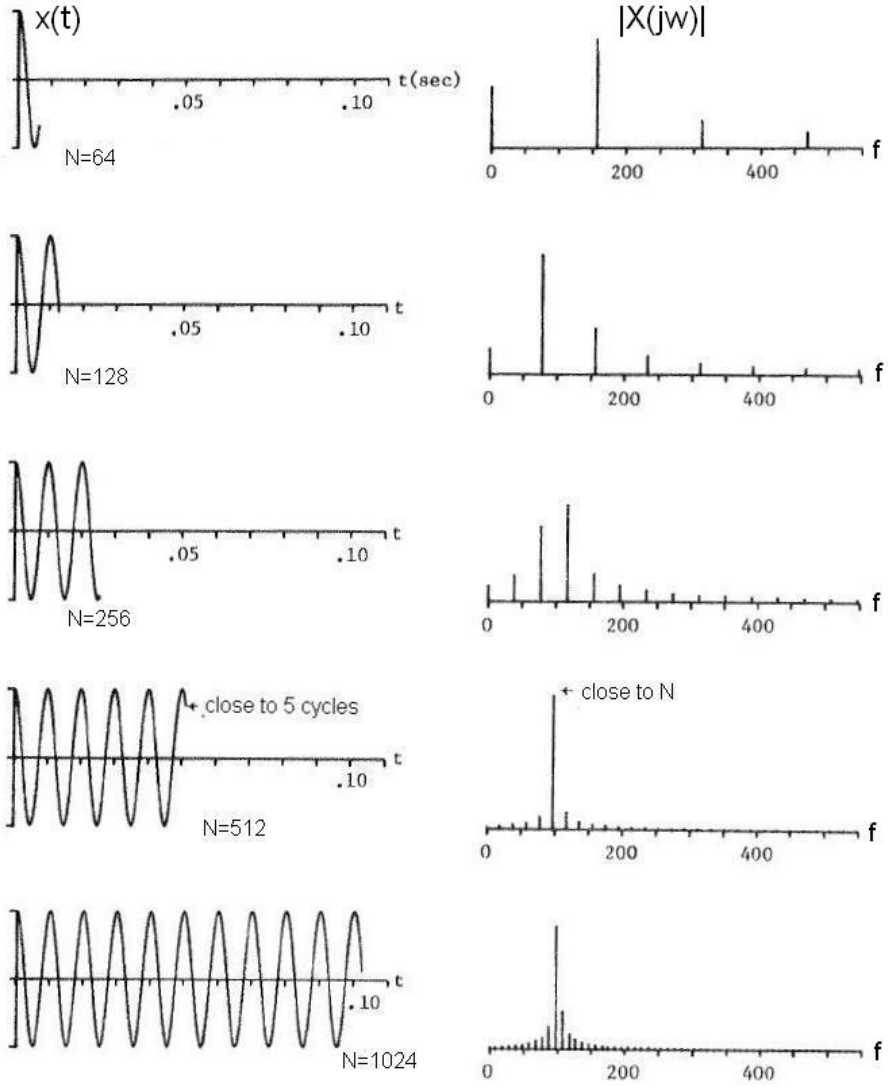

Figure 5.15.: Effects of different N on the spectrum computation of a sine with $f = 100\ Hz$ using the DFT [Ste79].

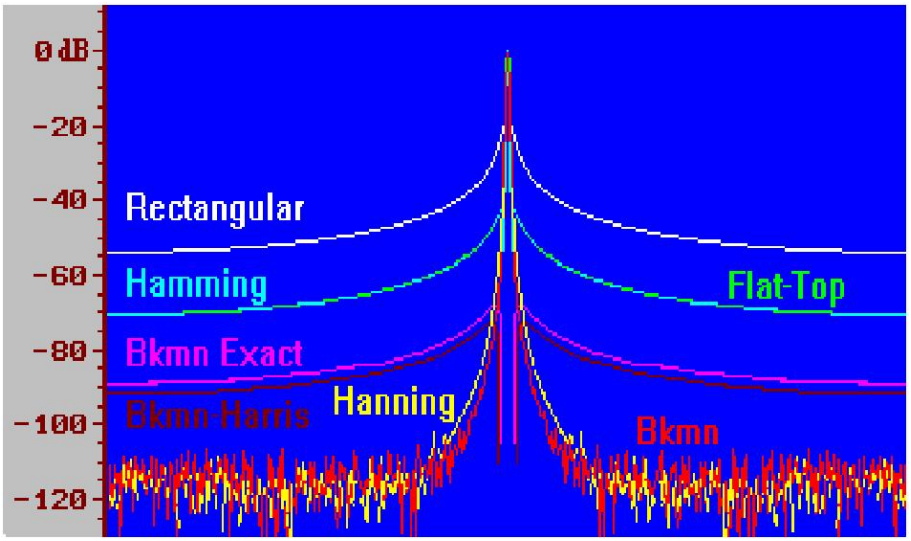

Figure 5.16.: Spectrum of different window functions [Res08].

Because of this the Hanning window was chosen, its definition is as follows:

$$w(t) = \frac{1}{2}\left(1 - cos\left(\frac{2\pi t}{NT_S}\right)\right); \quad 0 \leq t \leq NT_S \tag{5.31}$$

Figure 5.17 compares the characteristics of the rectangular and the Hanning window in the time and frequency domain.

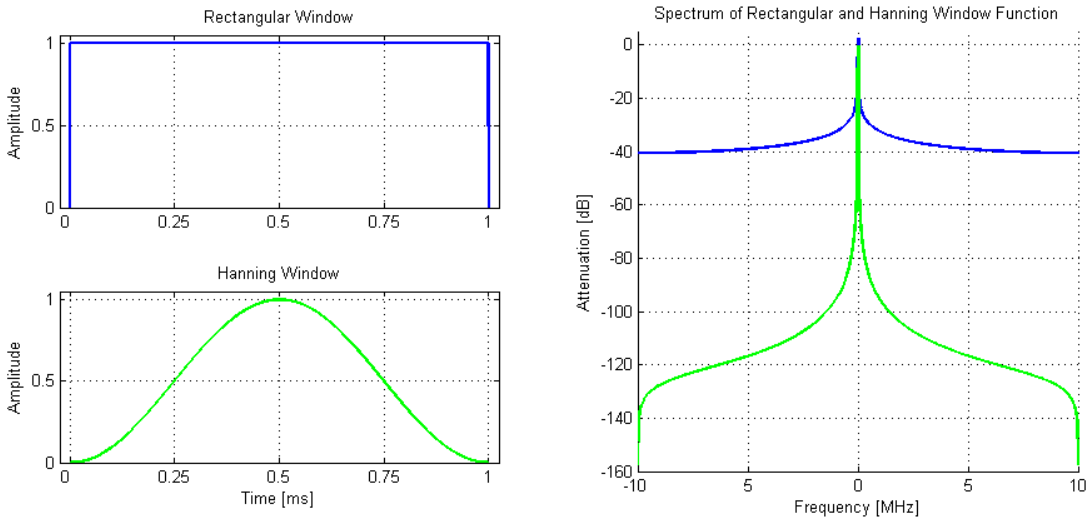

Figure 5.17.: Time domain plot and spectrum of rectangular and Hanning window.

Observing this plot, the Hanning window can attenuate the leakage effect much more as the rectangular window and so one can suggest that even in the presence of strong

CW-interference, the leakage effect will be quite small so that these interference detection techniques presented in Section 5.2 will work quite fine.

In order to clarify this, assume a CW-RFI with a ISR of about 60 dB, and a carrier frequency that leads to maximum leakage, i.e. its frequency falls exactly halfway between lines of the discrete spectrum, this leads to a maximum spectral width of the CW-RFI at the spectral detection threshold. When the CW-frequency does not fall on such a line the width of the window spectrum will be smaller, meaning that there will also be less leakage.

For the illustration of the spectral leakage problem in case of dealing with strong CW-RFI the following considerations were done: The amplified GPS signal has a power of 0 dB according to this the noise floor is assumed to be about $\approx$ 20 dB here, so that using the Rayleigh CDF a spectral detection threshold was defined here at 20 dB for a rectangular window. When now strong CW-interference occurs, the spectrum curve of the rectangular window, as depicted in Figure 5.16, is lifted so that it gets a serious width at the detection threshold. Consider, the spectrum of the continuous Fourier transformation of the CW-RFI would only have lead to a Dirac pulse.

Because of this reason the Hanning window was chosen, when it is lifted about 60 dB it has a smaller width at the detection threshold, meaning that there is less leakage.

The simulations clarified the necessity of using windows in case of dealing with strong CW-RFI. As a anticipation of this, Figure 5.18 shows the computed discrete spectrum using either a rectangular or a Hanning window on a set of data with $NT_S = 1\ ms$, $N = 20000$, $CW - ISR = 60\ dB$ and $f_{CWoff} = 500\ Hz$ apart the intermediate frequency using a rectangular and a Hanning window.
Note, the detection thresholds are equal here, because the results of the FFT using a Hanning window were multiplied with a standard correction of 2 before displaying. This multiplication of 2 is necessary, because it concerns the energy loss that occurs when using a Hanning window (the energy loss is 0.5).

Last but not least it is important to mention that N, the number of samples in a sampling interval, is also a parameter for interference detection to concern, especially when it comes to the detection of pulsed RFI in the frequency domain, for which a detailed discussion is presented in [MR06]. However, in that paper the number of investigated FFT samples were quite low ($N = 16\ to\ 128$) and it was shown that the performance of the algorithm increases with larger N. In this thesis the FFT was generally computed on vectors with a length of $NT_S = 1\ ms => N = 20000$, so a detection of pulsed- and CW-RFI-sources works fine. The number of samples was

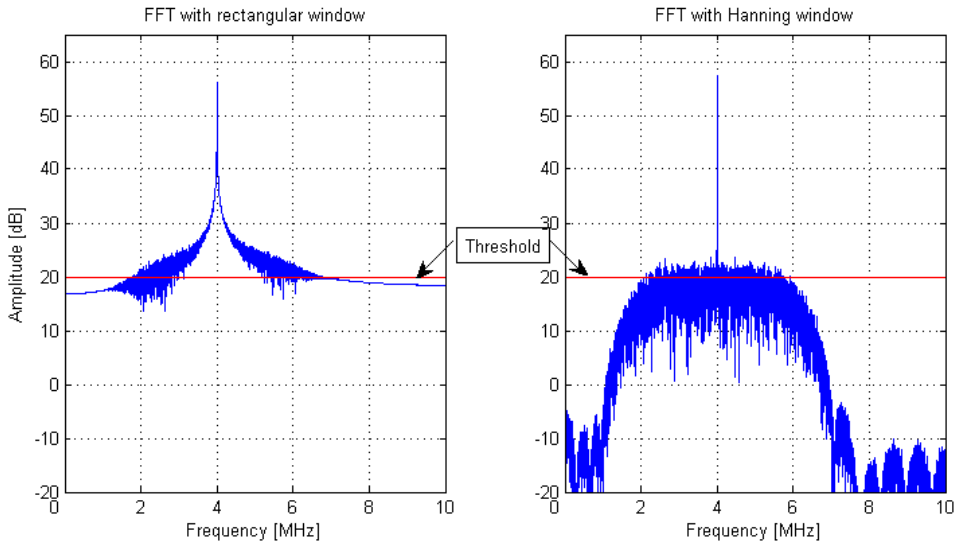

Figure 5.18.: Signal spectrum using the FFT with a rectangular- and a Hanning window.

set according to this value, because later this algorithm shall work together with an AGC, of which the gain is adapted every 1 ms.

5.5.5. Minimizing Window-Processing Loss

As depicted at the end of the last section, using window functions, different from the rectangular one, leads to a energy loss. In terms of signal power, this attenuation results in a 6 dB loss for the Hanning window. Going beyond the signal-power loss, window edge effects can be a problem when trying to detect short duration interference, such as pulsed RFI, that might occur when the window function is near its edges. The most common technique to minimize signal loss due to window edge effects is making use of overlapped windows. The strategy of overlapped windows is depicted in Figure 5.19. It is a straightforward technique where a single good window function is applied multiple times to an input data sequence. Figure 5.19 shows an N-point window function applied to the input time series data four times resulting in four separate N-point data sequences. Next, four separate N-point FFTs are performed, and their outputs averaged. Notice that any input sample that's fully attenuated by one window will be multiplied by the full "gain" of the following window. Thus, all input samples will contribute to the final averaged FFT results, and the window function keeps leakage to a minimum. Figure 5.19 shows a window overlay of 50 % where each input data sample contributes to the results of two adjacent FFTs. It is not uncommon to see an overlap of 75 % being used where

each input data sample would contribute to the results of the three individatal FFTs [Lyo01]. Of course the 50 % and 75 % overlap techniques increas the amount of total signal processing required, but when it comes to the frequency excision techniques, making use of overlapped Hanning windows instead of non overlapped windows will lead to a better C/N_0 of the received GPS signal, as shown in Chapter 7.

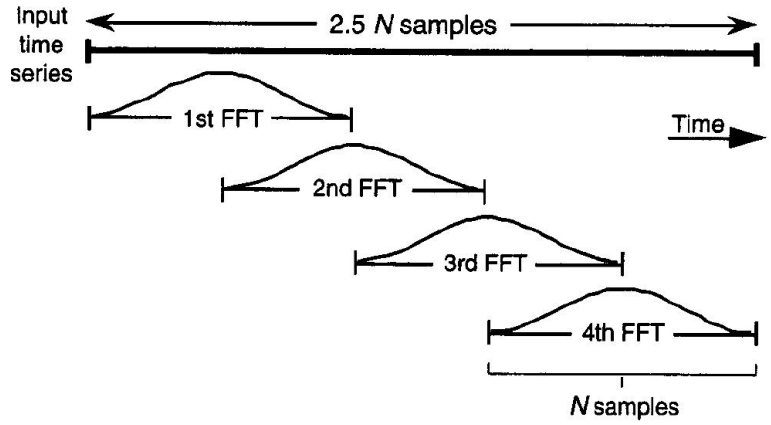

Figure 5.19.: Windows overlapped by 50% to reduce windowed-signal loss [Lyo01].

6. Interference Mitigation Techniques

Generally, the GNSS, which uses spread spread-spectrum techniques, already has a high robustness against any kind interference. Especially when it comes to wide band interference, that can only be rejected with the signal's spread-spectrum properties.

But however, the ability of interference suppression with those spread-spectrum techniques only is limited, what for this Chapter presents pre-correlation interference mitigation techniques.

These techniques presented here, work in connection with the interference detection techniques presented in Chapter 5. So they can take play either in the time or frequency domain. In this context temporal blanking and frequency excision techniques are proposed, using the quantized samples of a uniform ADC.

A different interference mitigation technique arises when using a nonuniform ADC. In this case, an AGC steers the input signal power according to the ADCs quantization levels so that only using that AGC/ADC a CW interfere can completely be rejected, see Section 6.1.

However, this technique proposed by Amoroso is quite old, meaning that to the time of proposal only ADC supplying little number of bits were available.

But due to technological advance, nowadays high multilevel ADCs are available that do not make it necessary to implement that gain steering method. In this work, the AGC's only task is to avoid saturation effects in the ADC.

Note: There are several other interference mitigation techniques than presented here, such as digital beamforming, adaptive Notched filters and post-correlation interference mitigation techniques. Those techniques discussed here are all of pre-correlation manner and depend on the digitized samples.

6.1. Nonuniform Adaptive A/D-Conversation

In [Amo83] Frank Amoroso gives a description for a nonuniform A/D conversion law to mitigate the impact of CW interference. But in order to understand this interference technique one need to understand the nature of the signal in presence of CW interference first. Therefore consider the incoming signal at IF, where a PRN-sequence mixed with a CW interference occurs. For illustrative purpose, Figure 6.1 shows a CW interference offset from the chip rate by an amount equal to 1/32.

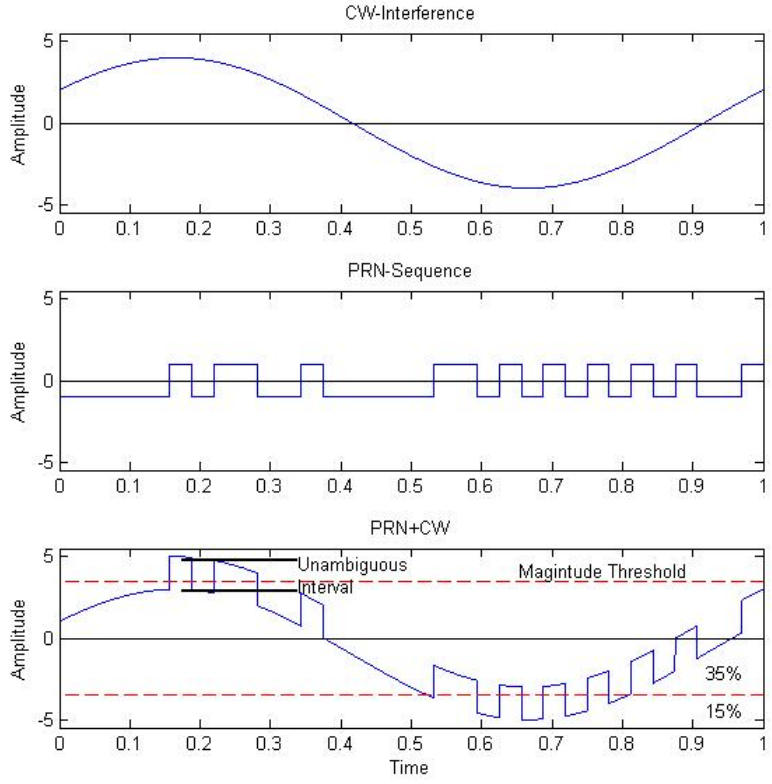

Figure 6.1.: Threshold strategy in CW interference, $SNR = -12\ dB$.

The first plot of Figure 6.1 shows only the CW-envelope, the second the PRN-sequence and the last the PRN-sequence mixed with the CW interference. As one can see at the A/D-converter input the PRN-sequence appears with a slowly fluctuating interference component. The interference undergoes on complete cycle in a 32 chip span. That interference envelope has an amplitude of 4 times that compared to the signal amplitude. Therefore the SNR is $-12\ dB$. The ADC used in [Amo83] produces 2 bits per chip: a sign bit and a magnitude bit. The threshold for the sign bit is at 0, the threshold for the magnitude bit are the dashed lines.

In order to mitigate the impact of interference Amoroso proposed two methods. The first method is to exclude all samples which fail to pass one or the other magnitude from the correlation process. All those with sufficient magnitude to pass a respective threshold are given the correct sign and equal weights in the correlator. From Figure 6.1 it is clear that every chip decision so entered into the correlator is correct. Every chip which passes the positive magnitude threshold is a positive chip, every chip that passes the negative threshold is a negative chip. The data correlation process will integrate these chips to make a perfect channel bit decision. In the situation of Figure 6.1 about 15 % of teh chips pass the upper threshold and 15 % the lower one. As a result, about 30 % enter the correlator. As long as there are enough chips per channel bit that at least a few chips pass a magnitude threshold then the CW interference will completely be rejected.

The adaption strategy in [Amo83] consists in exercising statistical control over the thresholds so that the threshold are exceeded with the required statistical frequency. For instance, in the situation of Figure 6.1 the requirement is that the sign threshold partition the samples equally, 50 % above and 50 % below. Also, 15 % of the chips should fall above the upper magnitude threshold and 15 % below the lower magnitude threshold, as indicated in Figure 6.1. Statistical counting combined with feedback is used to keep the threshold set properl yin the presence of circuits drifts, changes in signal level, changes in CW interference level, etc. [Amo83].

The second strategy to mitigate the impact of CW interference presented in [Amo83] is to include also those samples which failed to exceed one of the magnitude threshold. This approach is useful because in the presence of Gaussian noise the above described algorithm will lead to a poor performance. So in a less drastic strategy, in which the high magnitude samples are given much more weight in the data detection process than the small magnitude samples, a excellent performance for the suppression of CW interference in presence of Gaussian noise is achieved. For a detailed description of CW interference mitigation for nonuniform quantization law see [Amo83].

In the following the IMT do not depend on nonuniform quantization laws, due to the fact that to the time when [Amo83] was proposed in 1983, just ADCs with little quantization intervals were available in the case of [Amo83] there were four quantization intervals only. Nowadays available ADCs have much more quantization intervals, the one used for simulation and analysis in this thesis has 14 bits in total which is equipollent with having 2^{14} quantization intervals. For thus a digitized signal with such a high resolution, there are different interference mitigation strategies which appropriate higher robustness in presence of strong interference compared to that proposed in [Amo83]. The following sections deal with these IMT.

6.2. Temporal Blanking

As is know, the GNSS L5 band suffers under the impact of pulsed signals, such as DME/TACAN. The initial proposal on how to cope with these signals was to implement a pulse detection and blanking circuitry. The first circuitry proposed was using analog technology, but a digital solution was presented later in [Gra02]. This section depicts that temporal blanking technique, using those temporal pulse detection techniques presented in Section 5.3.

As proposed in [Gra02], digital pulse blanking is implemented using an ADC having more bits than required in the digital tracking loops. In this published work, 8 bits were used to quantize the signal but the signal was eventually represented using only 3 of those 8 bits for further digital processing. The additional available bits were used to implement digital blanking.

The principle of the digital pulse blanking is simple. It relies on the fact that pulses are short, about 3 μs, and have very large amplitude as compared to the noise level. Each sample is quantized over the all available bins and its quantized value is compared to a threshold. If the value is above that threshold then the sample is set to zero as a pulse is declared to be present.

The detection threshold as indicated in Section 5.3 is chosen as a compromise between the ability to detect pulses an the C/N_0 degradation in the absence of pulses. It is important to recognize that depending on the noise standard deviation and the value of the threshold, there is a certain probability that valid samples are above the threshold so a portion of the useful signal will be blanked. Thermal noise is also lost but signal lost dominates. The SNR degradation is indicated, in this case, in [Gra02]. Obviously, the AGC gain setting should not take into account blanked samples because nominal operation of this system is in presence of thermal Gaussian noise only. Samples prior to digital pulse blanking may contain pulses and samples right after blanking include zeros. The AGC gain setting would be respectively too low or too high [Bas03]. Figure 6.2 shows the digital blanking circuit that was implemented in this work.

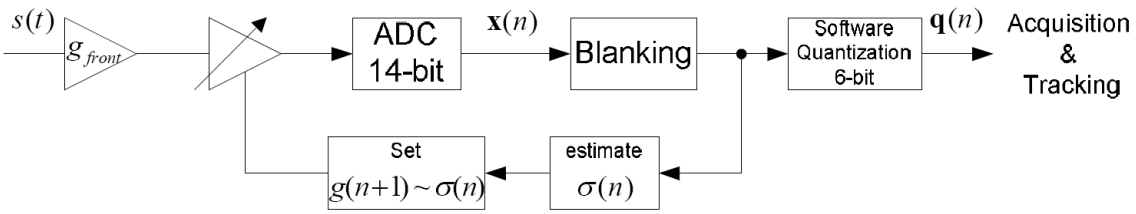

Figure 6.2.: AGC steering mode when digital blanking is implemented.

As described above, the method proposed in [Gra02] and Section 5.3, sufferers under the lack that it cannot identify the beginning and end of a pulse. But when the structure is modified, so that the detection takes place by monitoring energy fluctuations in a register, as shown in Figure 5.4, only those samples clearly belonging to a pulse will be blanked. Another benefit that arises is that this technique also allows to blank those samples directly before and after the pulse start and end detection, as they are also corrupted by them. As simulations showed, the achieved C/N_0 for that detection and mitigation technique work slightly better as the on presented in [Gra02].

A question that remains is, how much the signal at the correlator output is degradated due to blanking. For a description in detailed see [Bas04]. Further one can find there a investigation of what problems of pulse detection arise, using the in [Gra02] proposed method, when pulse collisions occur, which are not concerned in this work.

6.3. Frequency Excision Techniques

In Section 6.2 it was shown how to remove temporal power fluctuations using a technique working in the time domain. The issue of this technique is that it cannot be used to reject narrowband RFI such as CW.

A description of how to detect it in the PSD of the incoming signal is to find in Section 5.5, where the FFT was introduced in order to detect power fluctuations in the frequency spectrum. According to this a spectral-threshold was defined. All those frequency bins which exceeded this threshold were treated as interference.

On the basis of this detection strategy there are two possible interference mitigation techniques in general. One can either make use of adaptive Notch-filters or frequency excision techniques on the computed discrete spectrum. This Section presents those frequency excision techniques and what one needs to consider when strong RFI occurs.

6.3.1. Frequency Excision Techniques without an AGC

The great advantage of frequency excision techniques is that they can mitigate the impact of CW, narrowband as well as pulsed interference on the correlator loops. In

the following its operation mode is explained on the basis of Figure 6.3 where the developed interference detection and frequency excision circuit is depicted.

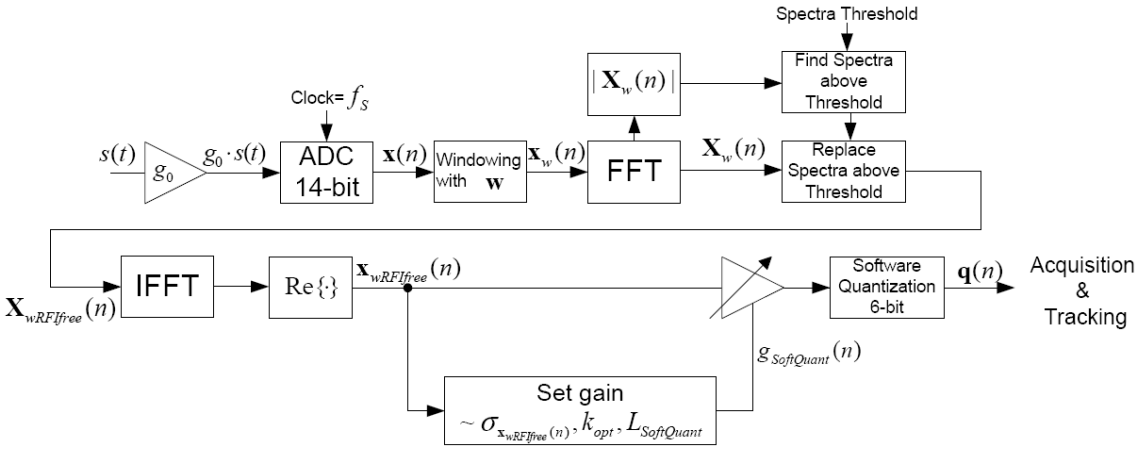

Figure 6.3.: Frequency excision circuit design to mitigate the impact of CW, narrowband and pulsed interference.

The input signal $s(t)$ labels the front-end-output signal, which is here the amplified, band limited and downmodulatet to an intermediate frequency antenna output signal. In the first step this signal is amplified with a constant gain g_0, which was set according to k_{opt} from table 3.1, to guarantee minimum losses due to the quantization, for the RFI free case. As one can see the ADC has 14 *bits*, which is quite much. So k_{opt} is not the ratio of the maximum ADC level to the input standard deviation, no, it is the ratio between the maximum quantization threshold of the initially used bits, which is at least the number of bits necessary for further digital signal processing (6 *bit* in this thesis), and the standard deviation of the signal for the RFI free case. The remaining 8 *bits* are necessary to increase the dynamic range of the ADC and so the power of the frequency excision technique, see Section 3.2.

In the next step sampling with a frequency of f_S and a 14 *bit* A/D-conversion takes place. For simplification of circuit design, its output signal is a vector $\mathbf{x}(n)$, that contains the quantized data of a time interval NT_S ($T_S = 1/f_S$).

After that, the vector $\mathbf{x}(n)$ is multiplied with a weighting window $\mathbf{w}$ of length N, in order to reduce the spectral leakage effect, which cannot be neglected when the signal $s(t)$ is corrupted by a strong narrow band interferer. In the developed circuit this weighting window $\mathbf{w}$ can either be of rectangular or raised cosine shape.

Due to the energy loss that occurs when using a raised cosine window (a Hanning window in this thesis), the program code was developed so that one can decide

whether to neglect this energy loss and the problems of pulse detection when the pulse occurs to the edges of a window and use that simple Hanning window or one can reduce the energy loss and detection problems by utilizing overlapping Hanning windows.
The theory of using different windows is discussed in 5.5.

Next the transformation of the time vector $\mathbf{x}_w(n)$ in the frequency domain is performed using a N sample FFT, which leads to the frequency vector $\mathbf{X}_w(n)$, that has a spectral resolution of

$$\Delta f = \frac{f_S}{N}. \tag{6.1}$$

Computing the magnitude spectrum $|\mathbf{X}_w(n)|$ and making use of the interference detection algorithm presented in Section 5.5, interference is detected.

Next the frequency excision techniques take place, so all those FFT points which have violated the spectral threshold are blanked/zeroed.

The signal spectrum $\mathbf{X}_{wRFIfree}(n)$ is now interference free, so the next step is to do the IFFT to obtain the corresponding interference free sample time series $\mathbf{x}_{wRFIfree}(n)$. As not shown in this Figure, if one has chosen to use overlapped Hanning windows in the beginning, it is necessary to superpose this vector with the ones adjacent, so the energy loss due to windowing is avoided. Else if one has chosen to use a simple Hanning window a C/N_0 loss of $\approx 1.42\ dB$ occurs at the correlator output.

In the next step the signal $\mathbf{x}_{wRFIfree}(n)$ is amplified according to its current variance, k_{opt} and the maximum threshold $L_{SoftQuant}$ of a software quantizer. This is necessary for the subsequent digital tracking loops, which anticipate samples that are quantized with 6 *bit*. But due to having a 14 *bit* ADC and performing frequency excision techniques, the variance of the quantized samples $\mathbf{x}_{wRFIfree}(n)$ is different from that what it should be, but the greater problem is that the output type of the IFFT is complex and of type "double" due to blanking/zeroing some FFT points. So a software quantizer and a software AGC is necessary to "round" the values of $\mathbf{x}_{wRFIfree}(n)$ with minimum losses to fit the specifications, which were that the output data should be of integer type, owning 6 *bit* per sample, for the subsequent signal processing units.

6.3.2. Frequency Excision Techniques with an AGC

The last subsection presented a circuit using frequency excision techniques to mitigate the impact of interference. In the corresponding simulations this circuit showed an excellent performance when dealing with strong RFI sources, but its performance can even be improved when utilizing an AGC, too.

The AGC, that was developed in this context, is different from those discussed in Section 3.3.2. The task of the AGCs of Section 3.3.2 was to set the gain according to k_{opt}, which was the ratio between the maximum quantization threshold and the input standard deviation. When interference occurred, the signal was attenuated to fit k_{opt}, which was as shown in Figure 3.10 not the optimum gain to guarantee minimum quantization loss, because that ratio k_{opt} was valid for GNSS signals superposed with thermal noise only.

So, due to the fact that there is also a dynamic range of the ADC, the task of the AGC has changed to avoid saturation effects in the ADC only. This also brings the benefit that the pre-defined specifications, the output signal shall have 6 *bit*, are better fulfilled. Because when using the old AGC design, the gain decreases when the input power is increased immediately, so, the GNSS signal would be represented in less bits for even low interference powers, although saturation effects would not have been cause anyway, which lead to a worse signal quality.

Figure 6.4 shows the circuit combining the benefits of AGC and frequency excision techniques. As one can see it also includes a pulse detection device so that the gain might not be driven by pulsed like interference, the notations are as above, except $g_{front} = g_0 - g_{init}$ where g_{init} is the maximum amplification of the AGC.
Notice, in the final program code developed and presented in Appendix A.4, the pulse detection works on the basis of power fluctuations as introduced in Figure 5.4, because this technique can better distinguish between different types of interference. In this circuit the pulse detection unit is only used so that the AGC is not sensitive to pulsed RFI, so no temporal blanking takes place, although a hybrid consisting of temporal blanking and frequency excision techniques should supply the best performance.

Beside the problem of distinguishing between pulsed and other interference types in the time domain, another issue arises for the setting of the spectral detection threshold according to the current gain $g(n)$. When using only rectangular or Hanning windows, one has to weight the initial spectral threshold with $g(n)/g_{init}$ for the threshold adaption. Notice: Here a linear behavior of the ADC is assumed, i.e. one can perform a multiplication before or after the ADC and it leads to approximately the same results. This assumption can only be done when there is a high initial sig-

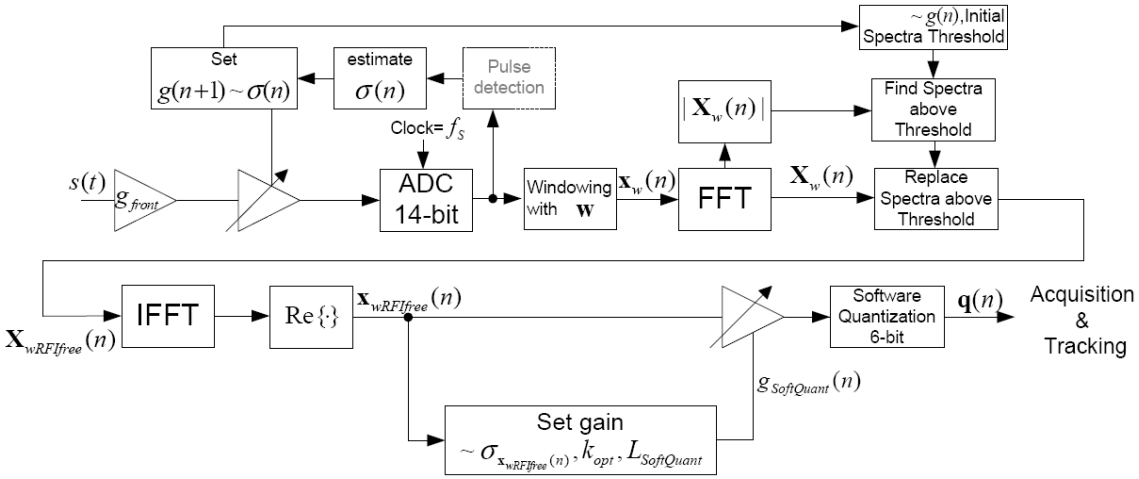

Figure 6.4.: Frequency excision circuit design with an AGC, to reject CW, narrowband and pulsed interference.

nal resolution and the AGC cannot attenuate the signal so much that quantization noise and hard limiting have do be concerned in the threshold detection adaption.

The up to now presented approach should be implemented for not overlapping windows, but when they shall be used problems arise, because two adjacent sets of data, i.e. two adjacent sample vectors, might have used different gains for the A/D-conversion and so there are different detection thresholds. This is due to the fact that the FFT of the overlapping window is computed on data belonging to one half to the vector $\mathbf{x}(n-1)$ and to the second half to the vector $\mathbf{x}(n)$. Due to the different gains there are some issues computing the FFT. Figure 6.5 illustrates the problem, when the gain is changes, the incoming signal will be weighted with different two different factors in two adjacent vectors, and the FFT of the overlapping Hanning window cannot be computed that easy.

In order to avoid the problem of different gains when computing the FFT of an overlapping Hanning window, the circuit design of Figure 6.6 needed to be modified, so that the gain change before the ADC is inverse multiplied with the ADC output vectors. The modified structure is depicted in Figure 6.6. The inverse multiplication of the gain after the ADC could be done here because the ADC owns a high resolution, like for the discussion on the threshold adaption strategy.

Notice: The multiplication of the inverted gain at the ADC output leads to the issue that the former ADC output values of type “integer” become “double”. In the Software defined radio this is no problem at all, but in FPGA design one might use the approach depicted in Figure 6.4 and not use the overlapping Hanning windows if the gain has changed, because excluding these overlapping windows from the overall

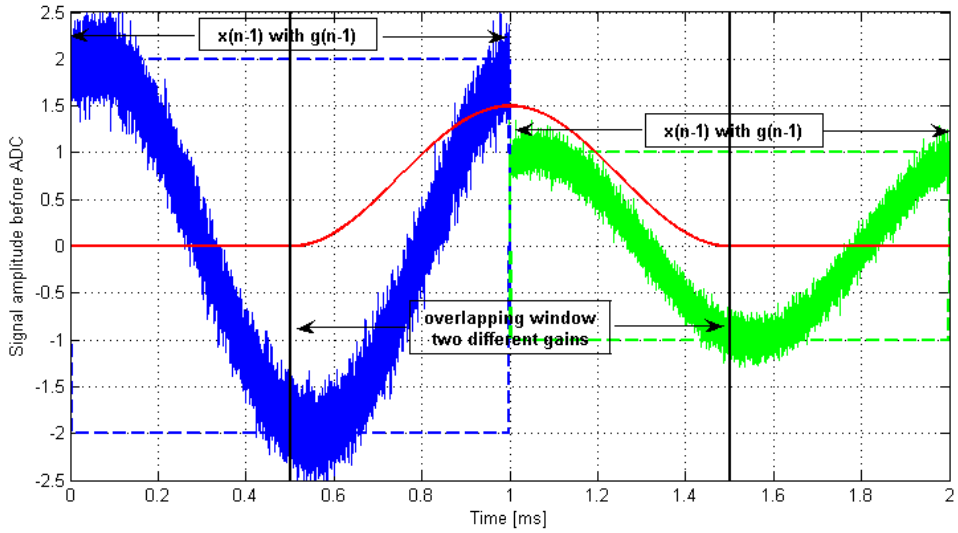

Figure 6.5.: Problem of FFT computation of overlapping windows when gain changes.

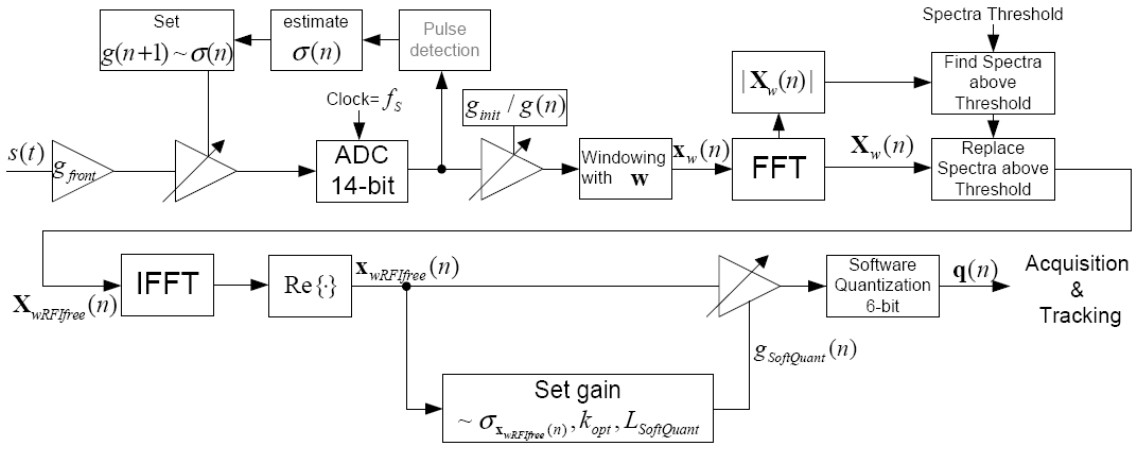

Figure 6.6.: Frequency excision circuit design with an AGC, considering the problem of FFT computation of overlapped windows, to reject CW, narrowband and pulsed interference.

computation leads only to a signal degradation of $\approx 1.42\ dB$ at the correlator but no to a loss of lock.

7. Simulations on Developed Pre-Correlation Interference Detection and Mitigation Techniques

This chapter investigates the performance of the developed interference detection and mitigation techniques. All signals that are examined here were produced with the DLR GNSS Software Signal Generator, which is presented first.

In order to evaluate the performance of the developed circuits it is necessary to know about the robustness of the GPS signal. Hence a detailed investigation about the GPS signal mitigating the impact of interference with its spread spectrum properties is given.

Proceeding from that, the performance of the different interference detection and mitigation techniques dealing with different kinds of interference was investigated.

7.1. The DLR GNSS Software Signal Generator

The DLR GNSS Software Signal Generator is a software approach to model various kinds of GNSS signal under the impact of RFI. Unfortunately the original program code did not distinguish between the "analog" and digital domain nor it was able to simulate pulsed RFI.

This Section gives a short insight into how the program code needed to be modified.

7.1.1. Modifications in the DLR GNSS Software Signal Generator

The former output signal of the DLR GNSS Software Signal Generator delivered already quantized samples. For this quantization the optimal ratio between maximum quantization threshold and input standard deviation ($k = L/\sigma$) was set according to the values obtained in Table 3.1 to guarantee minimum SNR degradation at correlator output. The issue is, that in order to set the optimal ratio $k = L/\sigma$ one needs to know the standard deviation of the analog input signal, which was available in the former signal generator. But for such high sampling rates which they are used in the real world DLR GNSS receiver, it is not possible to sense the standard deviation of the incoming signal in the analog domain, nor a the implementation of an AGC insensitive to pulsed RFI is ppossible.
Hence it was necessary to remove the AGC and ADC in DLR GNSS Software Signal Generator so that its output signal are "analog" samples of the received signal. Figure 7.1 shows the changed design of the generator output samples and Figure 7.2 shows as an example, the sampled signal in the time domain.

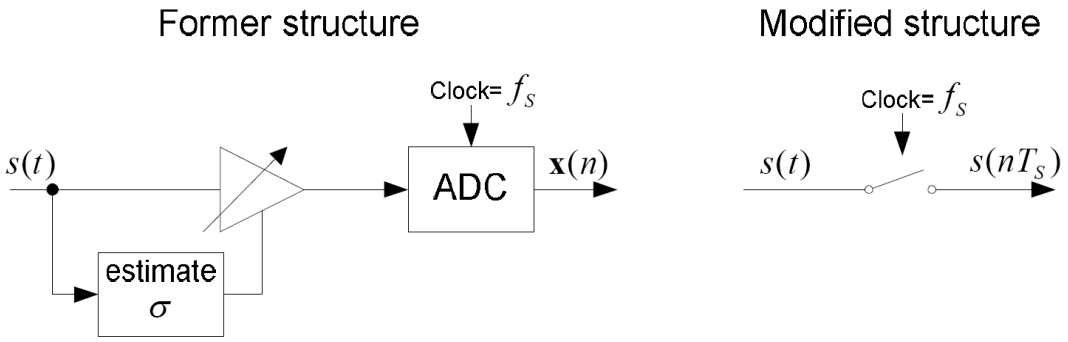

Figure 7.1.: Modifications in the DLR GNSS Software Signal Generator.

Furthermore, the former DLR GNSS Software Signal Generator was not able to simulate pulsed RFI, originating from sources such as DME or TACAN. So a program routine to produce DME pulse-pairs, according to Equation 4.4 was implemented. The parameters of that pulsed RFI, are as follows

- *Duty Cycle Time* (DCT)
- *Peak-to-Signal-Ratio* (PSR)
- Carrier Frequency

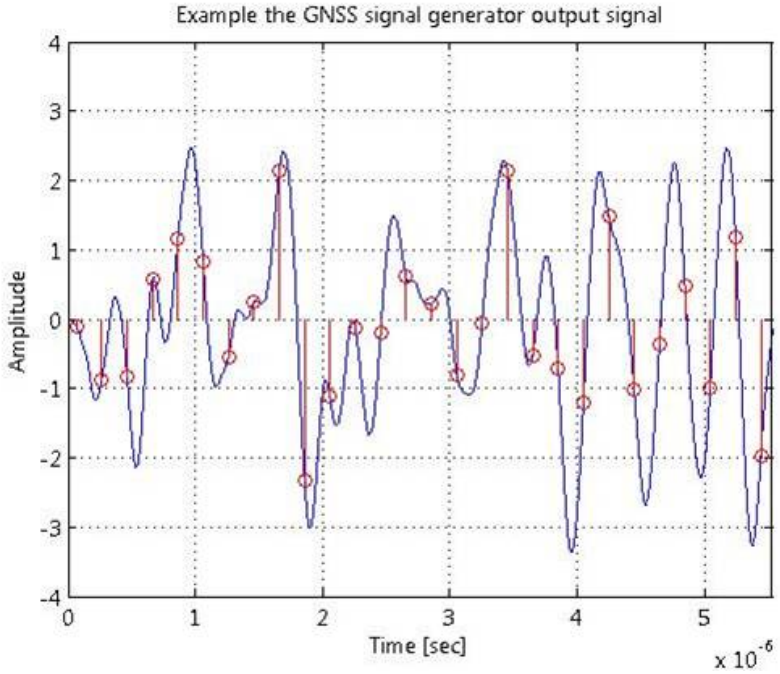

Figure 7.2.: Example of the sampled analog output signal of the modified DLR GNSS Software Signal Generator.

- *Direction Of Arrival* DOA

and can be changed in the generator file.

7.2. Simulation Parameters

All simulations were performed on the output data of the modified DLR GNSS Software Signal Generator. Thereby, the following listed settings were used:

- Carrier-to-Noise ratio:

$$\frac{C}{N_0} = 45\ dB \tag{7.1}$$

- Doppler Frequency:

$$f_{Doppler} = 0 \tag{7.2}$$

- Front-end-filter cutoff frequency:

$$f_{cutoff} = 2\ MHz \tag{7.3}$$

- Front-end-filter order:

$$N_{filter} = 32 \tag{7.4}$$

- Intermediate frequency:

$$f_{IF} = 4\ MHz \tag{7.5}$$

- Number of receiver antennas:

$$N_{ant} = 1 \tag{7.6}$$

- Sampling frequency:

$$f_S = 20\ MHz \tag{7.7}$$

- Satellite ID number:

$$ID_{Sat} = 3 \tag{7.8}$$

- Simulated signal type:

$$GPS - L1 \tag{7.9}$$

For the evaluation of the developed interference mitigation techniques the DLR GNSS Software Receiver version 5.31 was used. Its parameters were the same as for the generator and beyond that a threshold for loss of lock was defined as follows:

$$T_{LossOfLock} = 30\ dB. \tag{7.10}$$

7.2.1. Parameters for the Evaluation of the Impact of CW-RFI

A common method assess the robustness of a system is to continuously increase the power of a disturber, until the system is not able work any more. For this reason the DLR GNSS Software Signal Generator was used to generate a file, containing a sample chain, within that the CW-ISR was increased by 1 dB every 200 ms over a time interval of 28 s.

In Figure 7.3 the x-axis illustrates the point in time and the y-axis refers to the corresponding ISR, as one can see, in the first 200 ms no RFI is present. This is necessary for the estimation of some parameters, such as the received noise floor to set the front-end gain and the computation of some statistical detection thresholds in an assessment window as explained in Section 5.2.

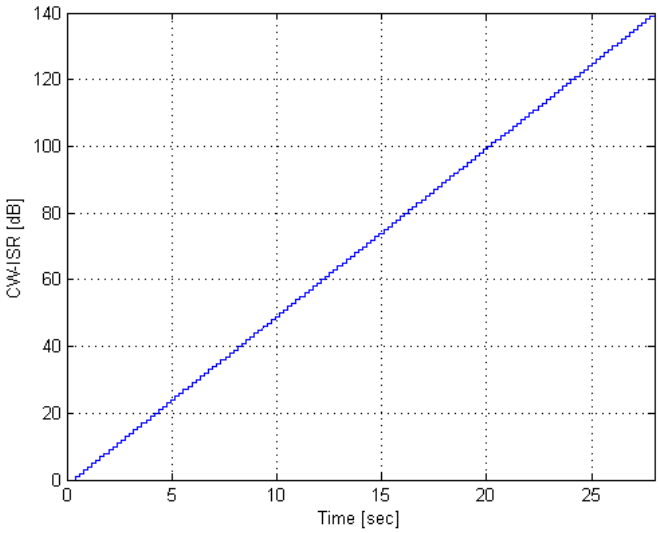

Figure 7.3.: CW-ISR versus time.

Another parameter to define was the CW-frequency. It was chosen in addiction to the temporal vector length of the ADC output, which is also a parameter of the AGC speed and the frequency resolution of the FFT. The time interval of that the FFT was computed had the temporal length of 1 ms, the sampling frequency was set to $f_s = 20\ MHz$, so each of those vectors had $N = 2 \cdot 10^4$ samples. This means that according to Equation 6.1, a frequency resolution of

$$\Delta f = \frac{f_S}{N} = \frac{20\ Mhz}{2 \cdot 10^4} = 1\ KHz \tag{7.11}$$

is obtained. From the theoretical discussion of the leakage effect in Section 5.5 it is known that it is worst for frequencies being halfway between the discrete frequency lines. Therefore the CW-frequency was set

$$f_{CWoff} = 500\ Hz \tag{7.12}$$

apart f_{IF}, so that the leakage effect is worst. For f_{CWoff} being a multiple of 1000 no leakage effect occurs. Figure 7.4 shows the computed discrete signal spectrum for a CW-ISR of 60 dB and $f_{CWoff} = 500\ Hz$ on the left and $f_{CWoff} = 1000\ Hz$ on the right.
As one can see for $f_{CWoff} = 1000\ Hz$ the discrete spectrum computation is correct because no spectral leakage occurs, whereas for $f_{CWoff} = 500\ Hz$ the computed discrete spectrum suffers strongly under the leakage effect.

The here described file was used for all circuit simulations of which have the objective is to detect and reject CW-like interference.

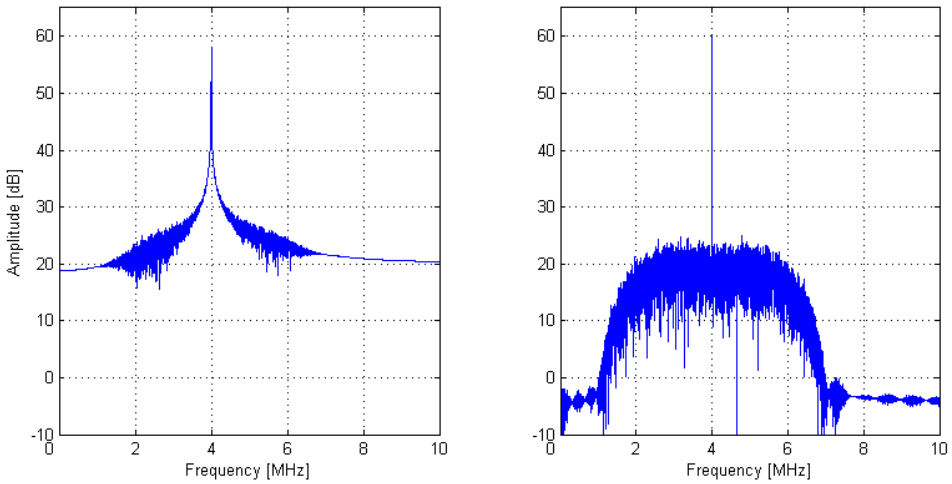

Figure 7.4.: Computed discrete signal spectrum for a CW-ISR of 60 dB and $f_{CWoff} = 500\ Hz$ on the left and $f_{CWoff} = 1000\ Hz$

7.2.2. Parameters for the Evaluation of the Impact of Pulsed-RFI

Similar to the generated CW-RFI file, also a data sheet containing a pulsed RFI source was generated. But note, due to the fact that all simulations were performed in the GPS-L1 band, it was necessary to simulate that DME also in that band, although in real world the GPS-L1 band is not corrupted by friendly pulsed RFI sources. This does not have an impact on the developed pulsed-RFI mitigation techniques as they do not depend on the frequency band. Only the obtained results for the GPS-L1 band should be worse than those for the Galileo E5a band, because the spreading codes of the Galileo have a higher robustness as GPS C/A code due to their greater length. For the generation of the GPS signal superposed with thermal noise and DME interference the following worst case parameters were chosen for an aircraft 10 km above a hotspot [MR06].

- Interference free time, necessary for the computation of some initialization parameters

$$t_{RFIfree} = 1\ s \tag{7.13}$$

- Simulation Time

$$t_{Sim} = 5\ s \tag{7.14}$$

- Duty-Cyle-Time

$$dct = 10\ \% \tag{7.15}$$

- Peak-to-Signal-Ratio

$$DME - PSR = 60\ dB^{1} \tag{7.16}$$

- Carrier frequency apart f_{IF}

$$f_{DMEoff} = 500\ Hz \tag{7.17}$$

7.2.3. Parameters for the Evaluation of the Impact of WB-RFI

For the generation of a WB-RFI, the DLR Signal Generator was used again, to produce a sampled "analog" signal in which the WB-ISR was increased by 1 dB/Hz every 200 ms over a timespan of 10 s. Figure 7.5 shows the ISR ratio versus time. The bandwidth of this WB source was chosen to be at least as large as that of the front end bandpass filter.

- WB-bandwidth

$$BW_{WB} \geq 2 \cdot f_{cutoff} = 4\ MHz \tag{7.18}$$

[1] For this value a thermal noise floor of $-205\ dBW/Hz$ was assumed. The SNR was computed with $SNR = C/N_0 - 10log_{10}(2 \cdot f_{filtEqBW}) = -20,47\ dBW$. The filter equivalent bandwidth was computed with the help of [REZ95, Page 677] and f_{cutoff}. This SNR computation is already available in the DLR GNSS Software Signal Generator. The noise power was computed to be $N_0 = -139,5\ dBW$. Above a hotspot a DME-PNR of $-102\ dBW$ is received [MR06], so finally the worst case DME-PSR could be computed ($DME - PSR = -102\ dBW - (-139.5\ dBW - 20.47\ dBW) = 58\ dBW$).

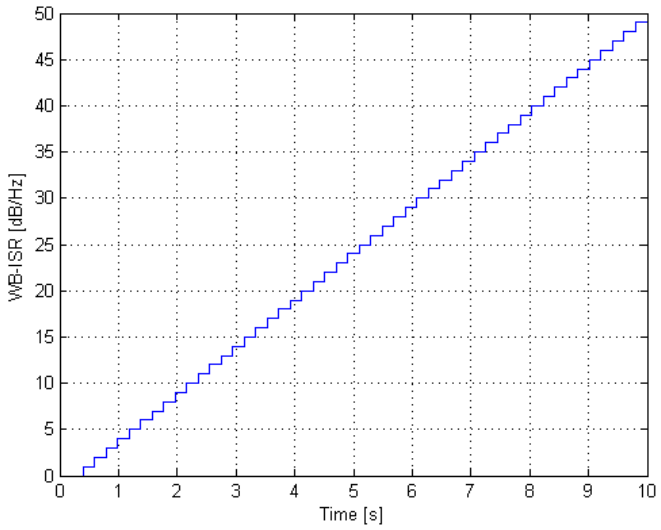

Figure 7.5.: WB-ISR versus time.

7.3. Simulations on the Chi-Square Test

From all the other detection algorithms presented in this work, the Chi-Square test is a little bit outstanding, because one can only detect interference but not use this knowledge to mitigate the interference neither in the frequency nor time domain. For the simulations, presented in this section, a 6 *bit* ADC, without a dynamic range, and an AGC using a look-up-table was used.

In order to use the Chi-Square test to detect interference the nominal bin loadings needs to be estimated first. Therefore Equation 5.9 was used with the maximum quantization threshold L normalized to one, and $\sigma = k_{opt} = 3$. As a validation of this equation, Figure 7.6 shows the theoretical bin loading (red curve) and the estimated bin loading (blue bars) using a 25 s long data file including a GPS signal superposed with thermal noise. The total available gain was set according to this ratio of $k_{opt} = 3$.

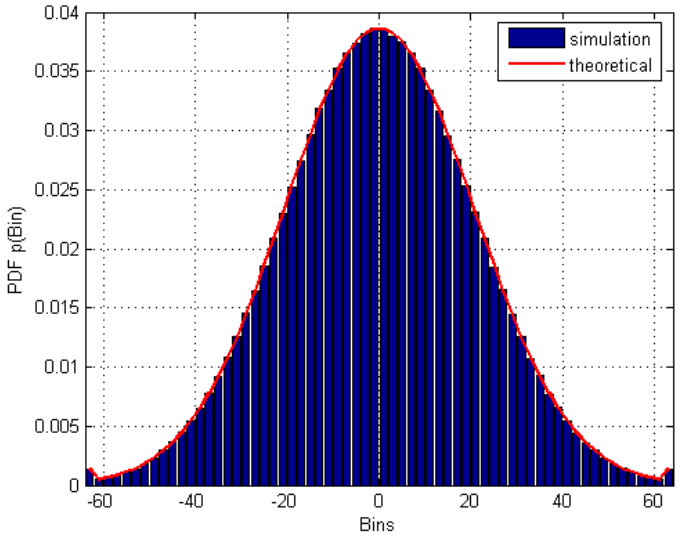

Figure 7.6.: Theoretical and estimated ADC bin loading.

Next, the Chi-Square Test was performed on that 25 s long data file where no RFI was present, Figure 7.7 shows the obtained distribution of $BinT$. According to Person's theorem [Kon08], that test statistic converges to the Chi-Square distributed random variable with $2^6 - 1$ degrees of freedom:

$$BinT \sim \chi^2_{63} \tag{7.19}$$

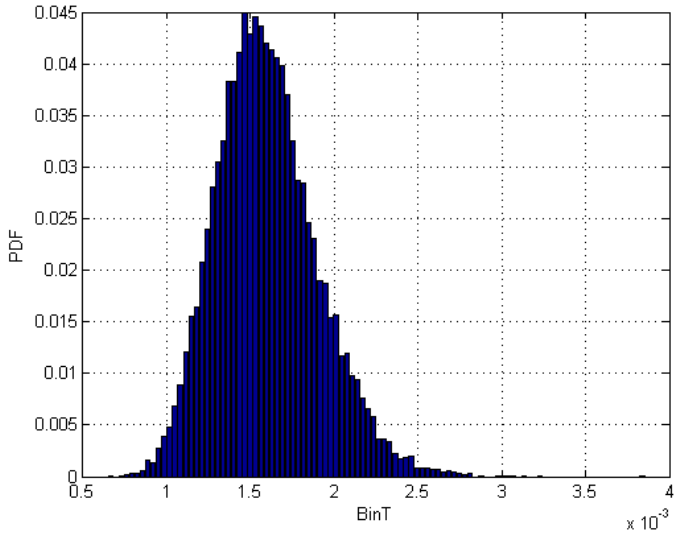

Figure 7.7.: Simulated Chi-Square test statistics in interference free case, for a 6*bit* ADC.

Proceeding from the obtained distribution, several values for $BinT$ statistics for achieving a specific false alarm rate are given in Table 7.1. According to this the detection of CW interference can be performed with the Neumann-Person test as defined in Equation 5.11.

False alarm rate	Threshold for $BinT$
0.01	0.002421
0.001	0.002770
0.0001	0.003112

Table 7.1.: Threshold values for a specific false alarm rate

As one can see, theoretical PDF fits the simulated ADC bins loading.

Next the Chi-Square test was performed on the file depicted in Section 7.2.1, in which the power of CW-interference is constantly increased. Figure 7.8 shows the obtained results. On the left one can see the gain adaption to ensure the optimum ratio k_{opt} and on the right the results of the Chi-Square test for different CW-ISRs are depicted.
Examining the plot, one can observe that detection of the CW-RFI is clearly possible using the Chi-Square test, but that RFI can even be detected earlier when monitoring the AGC which is the first device in the pre-correlation signal processing unit, or when initially using even more than 6 *bit* in the ADC to represent the signal.

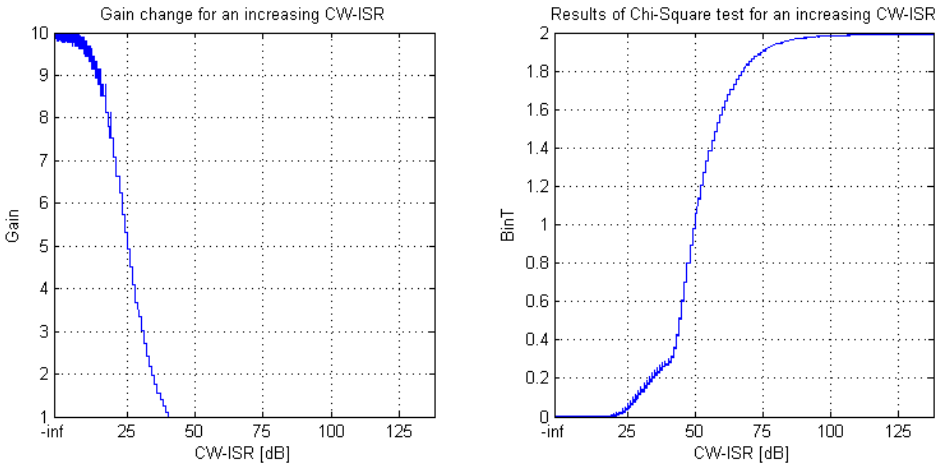

Figure 7.8.: Results of Chi-Square Test for an increasing CW-ISR.

In order to show the change in the Chi-Square distribution in presence of CW-RFI, Figure 7.9 illustrates the obtained Chi-Square distribution for a CW interference with an ISR of 25 dB. As one can see, the obtained distribution strongly differs from that for the noise only case, shown in Figure 7.7.

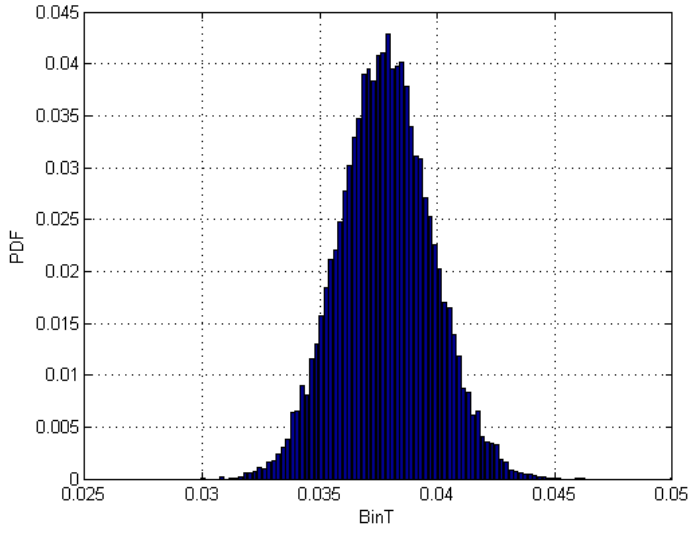

Figure 7.9.: Simulated Chi-Square distribution for a $CW - ISR = 25\ dB$

Up to this point the Chi-Square test was only performed on a signal having not a Gaussian amplitude distribution. Hence the possibility to detect WB interference exists, because ideal WB RFI owns also a Gaussian distribution of its amplitude,

like noise. Therefore the data file depicted in Section 7.2.3, which contains a WB interference with increasing power, was examined with the Chi Square test. Similar to Figure 7.8, Figure 7.10 shows the gain adaption versus WB-ISR on the left and the obtained values for $BinT$ on the right.

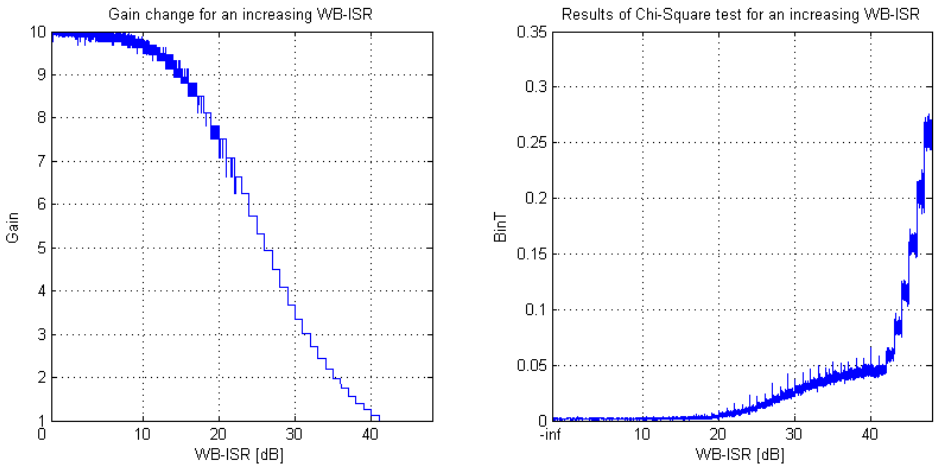

Figure 7.10.: Results of Chi-Square Test for an increasing WB-ISR.

Apparently an error occurs, because when the WB-RFI would have had a purely Gaussian distribution of its amplitude values, the results of the Chi-Square test in combination with the AGC should have stayed constant, as long the AGC is not in its lower border. Possible sources of this error are the signal production algorithm in the DLR Software Signal Generator and that when using a look a table the gain can only take certain values which might lead to a not optimal AGC steering. In order to exclude the error of the limited look-up-table resolution a pure WB-RFI signal with an ISR of 25 dB was generated and investigated. The results of this investigation are shown in Figure 7.11. It seems that the DLR Software Generator delivers for WB-RFI a not purely Gaussian distribution of its amplitude. Therefore red curve shows the theoretical and expected ADC bins loading, whereas the blue bars show the obtained distribution. To the time of proposal of this work the source of error in the signal generation could not be found.

Hence the results of the Chi-Square test differ from that for noise only. The obtained $BinT$ distribution is plotted on the left of Figure 7.11.

This section showed, that the Chi-Square Test is a valuable tool to detect interference, but by monitoring the AGC, interference can be detected even earlier.

Although this interference detection technique showed a good performance, it was not implemented in the final program code, due to its lack that it cannot estimate

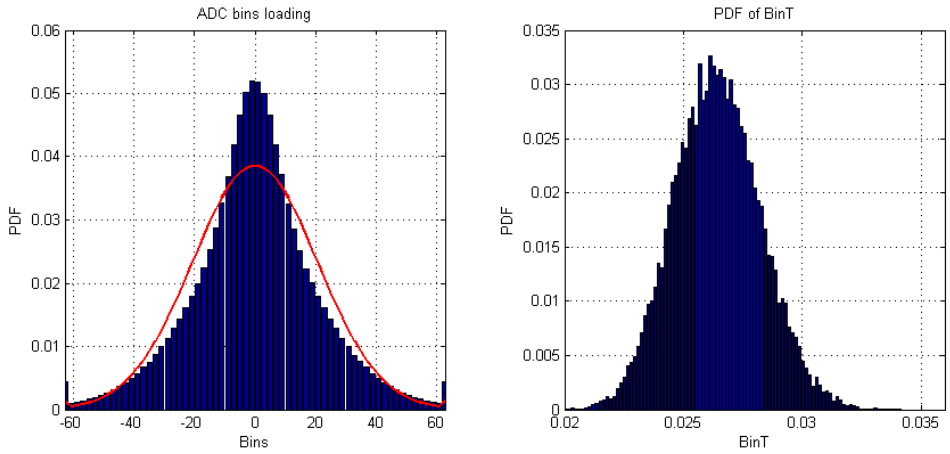

Figure 7.11.: Distribution on ADC bins and Chi-Square distribution for a WB-ISR of 25 dB

the specifics of an interference such as frequency or point in time when it occurs. Because of this issue more complicated techniques needed to be developed, which are able determine the specifics of the interference as shown in Chapter 5, so that those interference mitigation techniques presented in Chapter 6 can mitigate its impact on the correlator loops.

7.4. Simulations on the Robustness of GPS C/A Code

In order to assess the performance of the developed interference mitigation techniques, it is necessary to know first, up to what inference powers the GPS signal is able to reject interference alone with the spreading properties of the C/A-code. For these simulations, the circuit illustrated in Figure 7.12 was used.

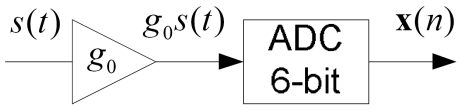

Figure 7.12.: Simple 6-bit ADC.

Here g_0 is a constant gain that was set according to the ratio $k_{opt} = 3$ between the maximum ADC threshold L, and the ADC input standard deviation $\sigma_{RFIfree}$. Further, a 6 *bit* ADC was implemented, because the pre-defined specifications were that the samples at the final circuit output should own a 6 *bit* resolution.

7.4.1. Robustness of the GPS C/A Code Dealing with CW-RFI

The investigations of the robustness of the GPS C/A code starts with a CW-interference. Therefore the CW-data file, depicted in 7.2.1, was driven to the ADC input.

First of all the impact of that interference on the ADC bins loading was examined, Figure 7.13 shows the histogram of the ADC bins loading versus the CW-ISR.

As one can see, for a CW-ISR of $-\infty$ dB the histogram has a Gaussian shape, because thermal noise is the prevalent signal (see also Figure 7.6 for the nominal ADC bin loading). But for the CW power increasing, the PDF of a sine wave, as depicted in Figure 4.6, becomes visible on the ADC bins and finally at a CW-ISR of $\approx 52dB$, the ADC is completely in saturation.

Next, having quantized that file with an increasing CW-interference power, it was evaluated in the DLR Software Receiver, to determine at which CW-ISR the receiver loses lock. Figure 7.14 shows estimated C/N_0 versus time at the receiver output.

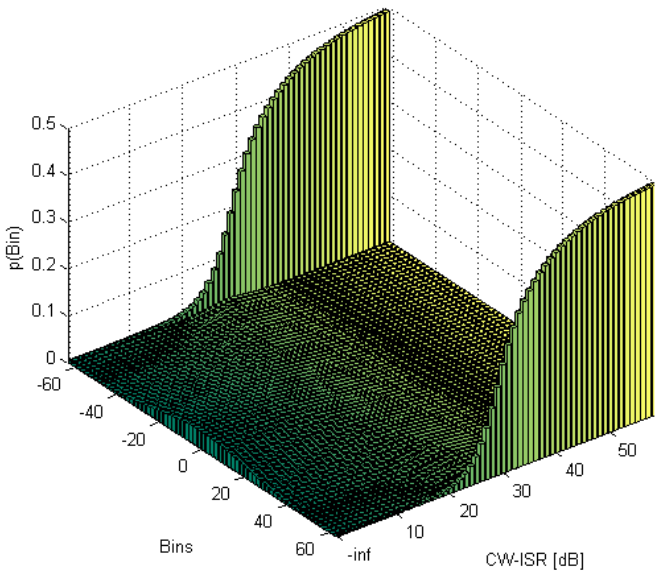

Figure 7.13.: ADC bin loading for a 6 *bit* ADC for different CW-ISRs.

One can observe that the receiver lost lock after ≈ 6.8 s of tracking, which equates a CW-ISR of 33 dB, see Figure 7.3 for a connection between time and CW-ISR. But fine estimations showed that the receiver does lose lock even earlier, at a CW-ISR of ≈ 29.5 dB. But notice, observing the correlator output presented in Figure 7.14, the receiver is able to deal with even higher interference powers, so when decreasing the loss of lock threshold defined in Equation 7.10 according to $T_{LossOfLock} = 30$ dB, the receiver should be able to proceed tracking for even higher interference powers, but this was not examined in this thesis.

Furthermore, for the loss of lock ISR, that was about 29.5 dB here, it is interesting to see, that the signal composed of CW-RFI, thermal noise and GPS signal, spend about 58.25 % of the time between the highest ADC bins, but the receiver was still able to proceed tracking, see Figure 7.15. Although one cannot make any statements with this percentage, it is in connection with the corresponding percentage for the frequency excision techniques interesting to know, because there it was much smaller, meaning that the ADC can be even more saturated for theses techniques to work properly.

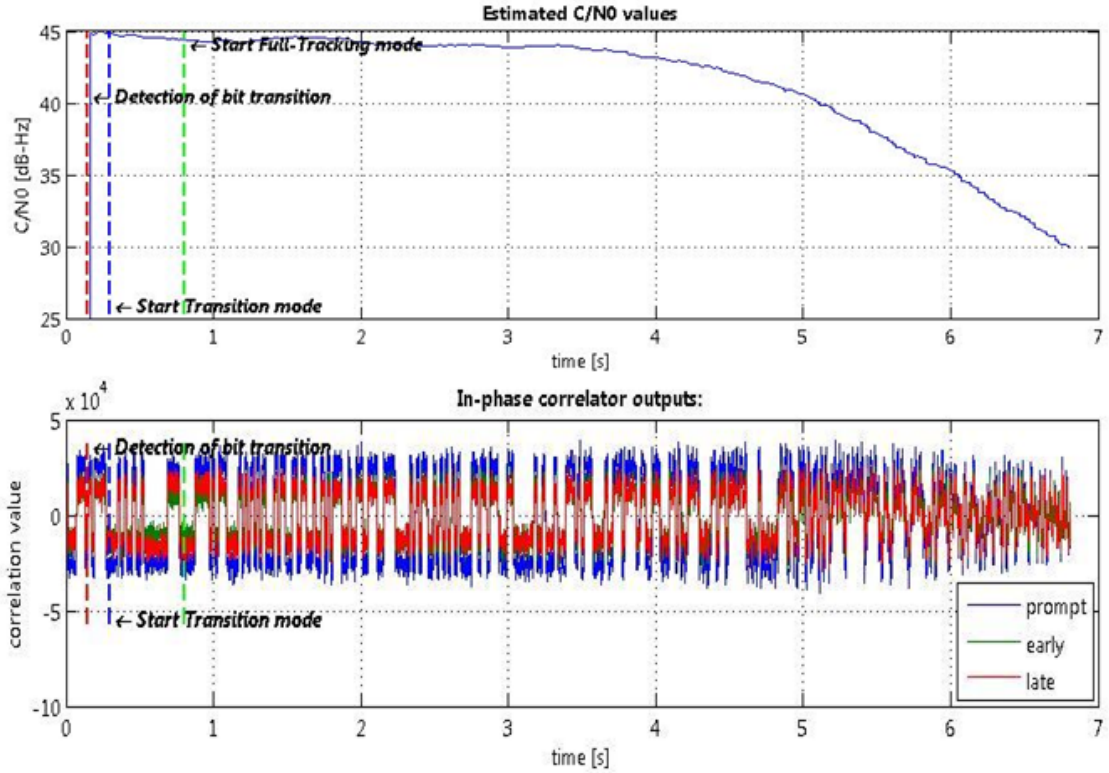

Figure 7.14.: Above: C/N_0 at the receiver output; Below: Correlator output; both for a 6 *bit* ADC under the impact of a increasing CW-ISR

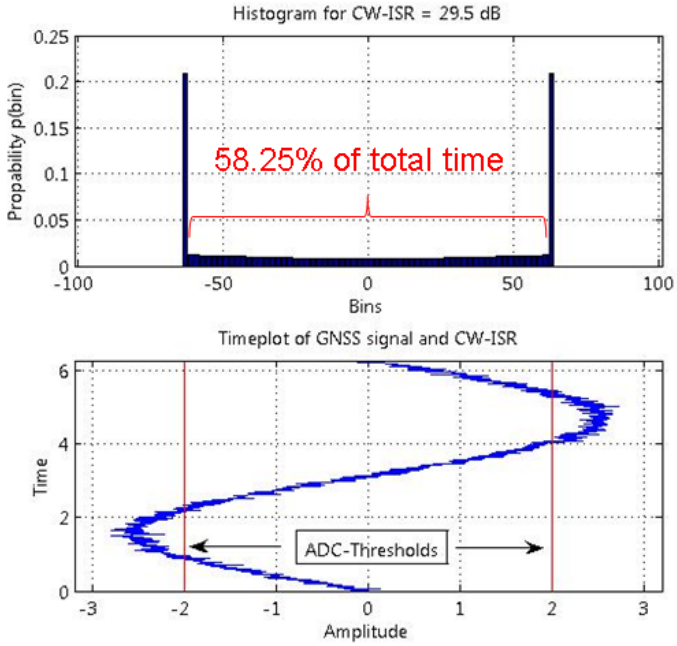

Figure 7.15.: Time in % that the signal spended between the maximum ADC bins to keep lock for a CW-ISR of 29.5 dB.

7.4.2. Robustness of the GPS C/A Code Dealing with Pulsed-RFI

Similar to the discussion in the last section, the circuit illustrated in Figure 7.12 was used to evaluate the robustness of the GPS C/A code dealing with pulsed interference. As pulsed interference source the signal depicted in Section 7.2.2 was introduced to the circuit's input. After the signal was quantized its quality was assessed in the DLR Software Receiver, Figure 7.16 shows the results.

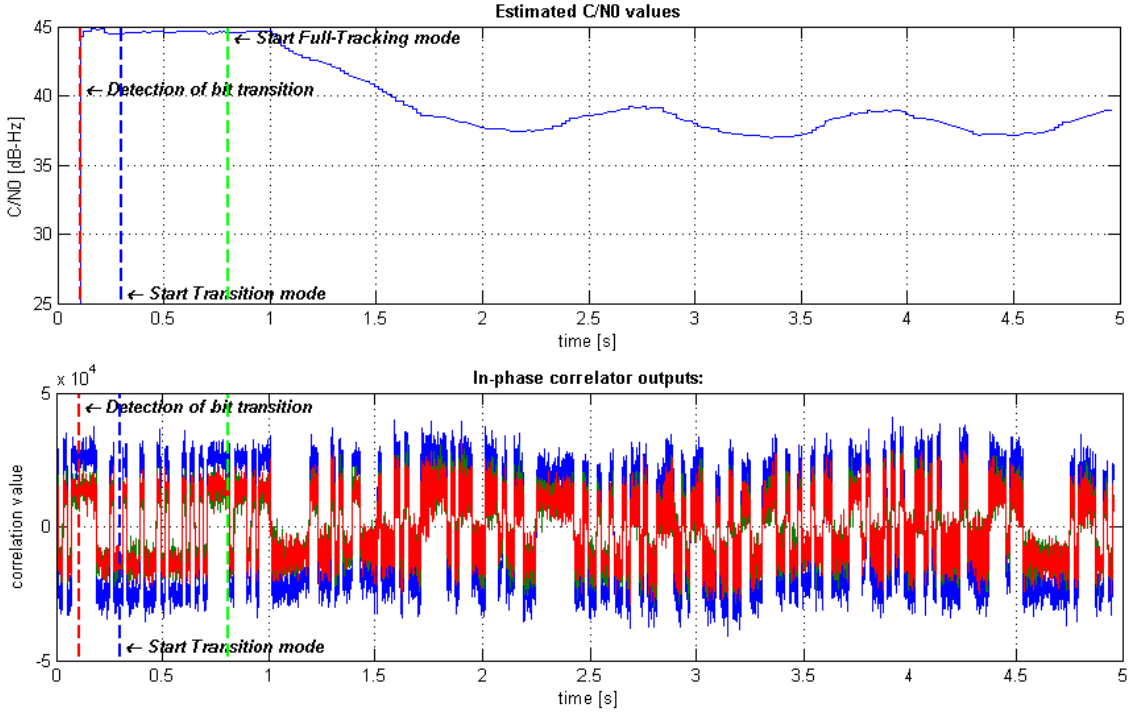

Figure 7.16.: Above: C/N_0 at the receiver output; Below: Correlator output; both for a 6 *bit* ADC under the impact of pulsed-RFI

As one can see, when the DME interferer is put on after 1 s of processing, the C/N_0 is decreased by about ≈ 7 dB, but this is no problem for correlator which is only little disturbed, so the receiver can proceed tracking without greater problems. But notice, the ADC here is ideal, meaning that even when it is strongly oversteered, as it is the case here, it has no time delay, nor it can be damaged. Hence, real world results might strongly differ depending on the specifics of the implemented ADC.

7.4.3. Robustness of the GPS C/A Code Dealing with WB-RFI

Wideband interference is the only interference type which can only be rejected by the signal's spread spectrum properties. But as mentioned before, the correlator needs digitized samples to perform the correlation. For the transformation of sampled "analog" wideband interference signal, depicted in Section 7.2.3, to the digital state the circuit illustrated in Figure 7.12 was used. Due to the fact that here is no gain adaption, saturation effects in the ADC occur, thus Figure 7.17 shows the histogram on the ADC bins loading versus WB-ISR.

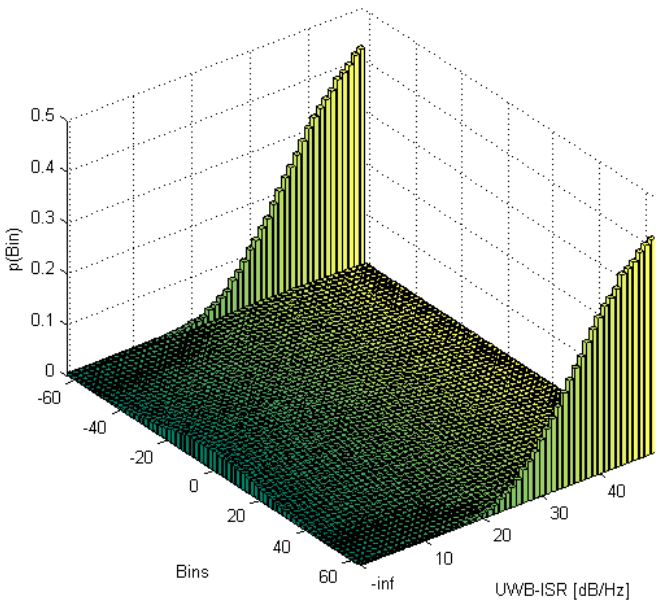

Figure 7.17.: ADC bin loading for a 6 *bit* ADC for different WB-ISRs.

Analyzing the plot, on can see that strong saturation effects occur, but anyway the receiver was able to keep track up to a WB-ISR of $\approx 43\ dB/Hz$, as illustrated in Figure 7.18.

But focusing on the correlator output, the signal quality of the GPS signal becomes quite bad after 8.4 s of processing, i.e. for a WB-ISR $\geq 40\ dB/Hz$, which was estimated to be the real loss of lock threshold.

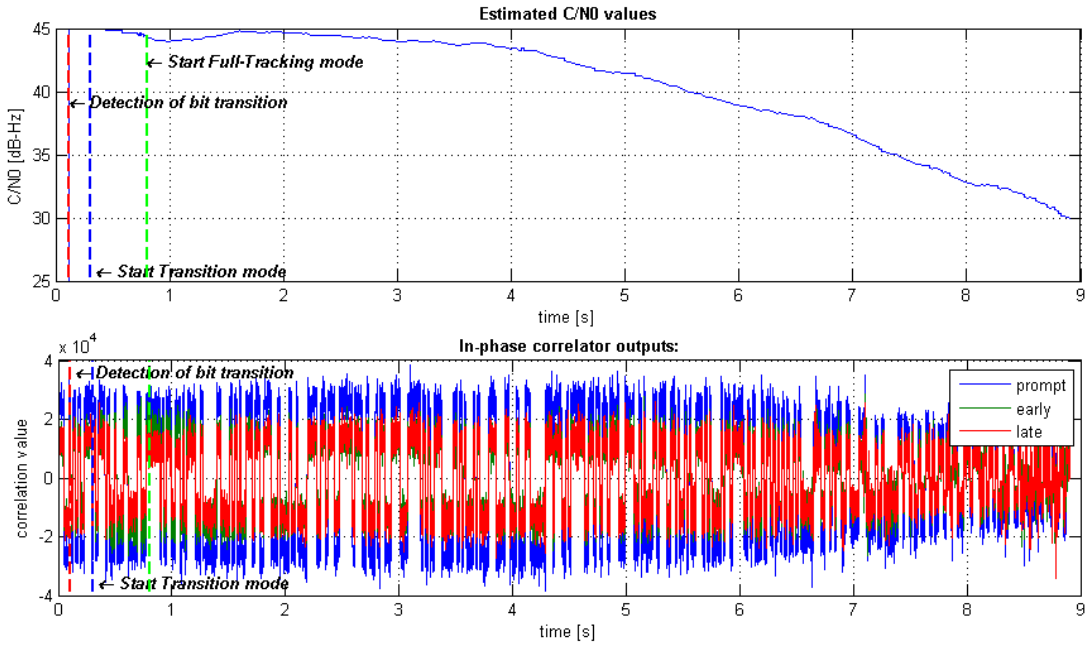

Figure 7.18.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 6 *bit* ADC under the impact of WB-RFI power increasing

7.5. Simulations on the Robustness of GPS C/A Code in Combination with an AGC

This section examines the performance of the GPS C/A Code in combination with a 6 *bit* AGC/ADC that respects the optimum ratio k, between the maximum quantization threshold L and the input standard deviation σ. The circuit used in these simulations is depicted in Figure 3.8, where a look-up-table is used to estimate the input variance on the basis of the ADC output variance. Like the simulations in Section 7.4, the ADC has 6 *bit* in total which are all initially used. So there is no dynamic range which would be necessary to detect pulsed-RFI and steer the gain correct. Because of that only the impact of an increasing CW-ISR and WB-ISR is examined here.

7.5.1. Evaluation of the Impact of CW-RFI on AGC/ADC

Like before, the data file depicted in Section 7.2.1 was used to investigate the robustness of the GPS C/A code dealing with CW-RFI, but this time using an 6 *bit* AGC/ADC for the A/D-conversion. The necessary look-up-table to estimate the correct gain on the basis of the ADC output variance was generated with the program depicted in Appendix A.1.

In Section 3.3 it was discussed that any increase of the signal input power leads to a decrease of the gain. This behavior of gain adaption could also be examined here. Figure 7.19 shows the gain versus time for that CW-RFI file.

As one can see, the gain decreases due to the increasing interference power, as expected. Gain fluctuations at the beginning originate from power fluctuations in the received noise floor, but when the CW-interference power becomes dominant, the AGC remains in a certain gain value of the look-up-table, due to the fact that the interference power remained constant in each 200 ms long step.

Next the ADC bin loading was examined for that CW-RFI source with increasing power. The histogram of the bins loading versus CW-ISR is depicted in Figure 7.20.

As one can see, due to the gain reduction the PDF of the mixture of sine and Gaussian noise becomes clearly visible on the ADC bins. Further one can examine that at time 8.2 s which is equal to a CW-ISR of 39 dB, the AGC is in its lower border. Hence for further increasing CW powers (CW-ISR$>$ 39 dB) the former compressed PDF of sine and Gaussian noise is spread over the remaining upper bins

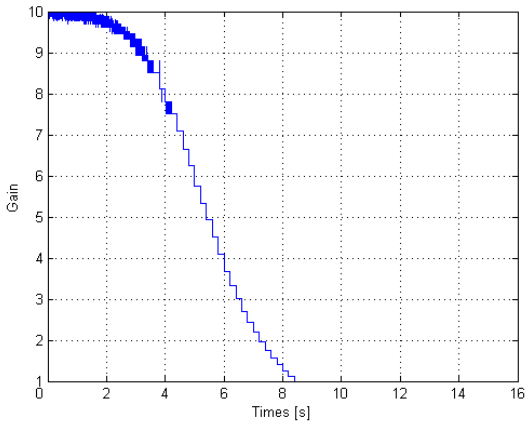

Figure 7.19.: Gain change for an increasing CW-ISR.

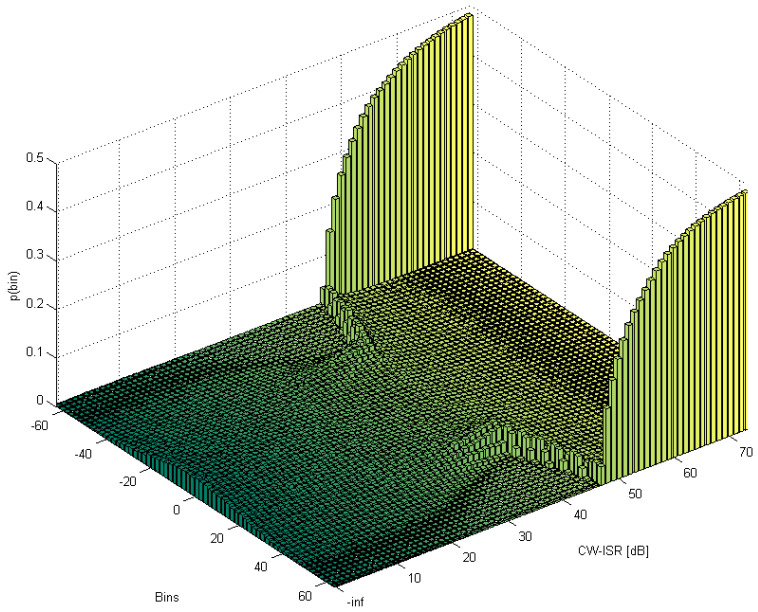

Figure 7.20.: ADC bins loading for a 6 *bit* ADC in combination with an AGC for different CW-ISRs.

and finally, at a CW-ISR of $\approx 72\ dB$ the ADC is completely saturated.

Notice, the AGC used in this thesis tries to maintain the optimum ratio k_{opt} between maximum ADC threshold and input standard deviation, which is only optimal for the RFI free case. Using a gain adaption strategy that respects the PDF of the interference on the ADC bins distribution would be better, because then the signal would be quantized over the whole ADC range always. But this a priori knowledge is not available for the AGC used in this work. Unfortunately a gain adaption strategy which automatically respects the PDF of any RFI source was not available.

But however, in contrast to the simple ADC depicted in Figure 7.12, the AGC was able to decreases saturation effects in the ADC. Former the ADC was for a CW-ISR of $\approx 52\ dB$ in saturation, now it is for a CW-ISR of $\approx 72\ dB$. This will bring some benefits in performing the correlation as shown in the following Figure 7.21, where the results of the DLR Software Receiver are illustrated.

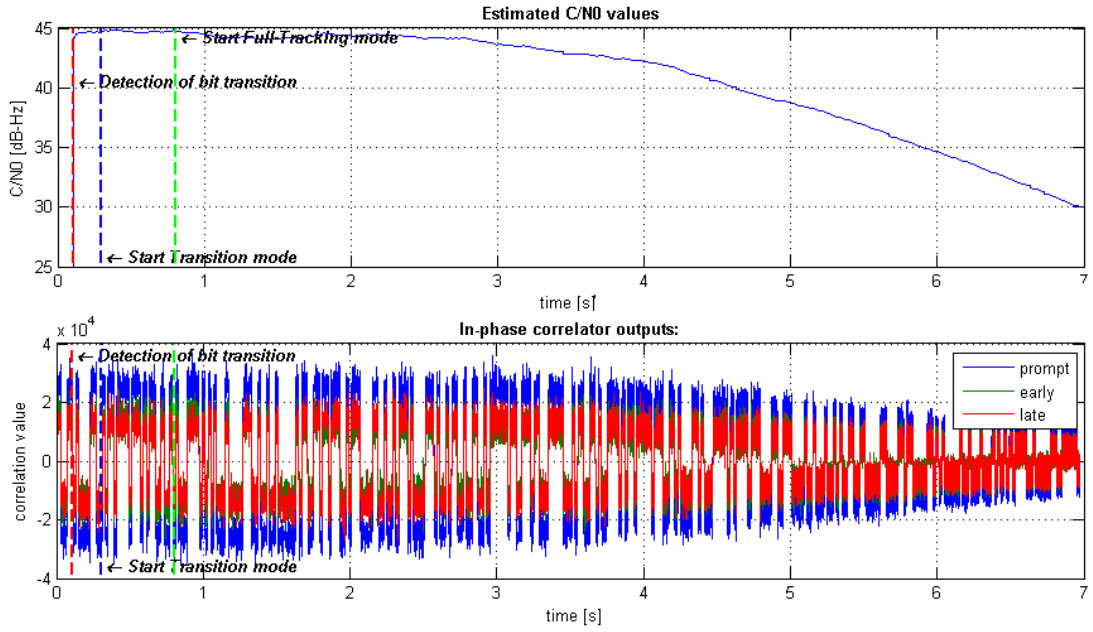

Figure 7.21.: Above: C/N_0 at the receiver output versus time; Below: Correlator output versus time; both for a 6 *bit* AGC/ADC under the impact of CW-RFI power increasing

Similar to the discussion in 7.4.1, for a larger loss of lock threshold, the receiver would have been able to proceed tracking for even higher CW-ISRs. But anyway, comparing the correlator output results of Figures 7.14 and 7.21 it catches ones eye that the signal quality has increased, although this gain adaption strategy was not optimal. Because when focusing on the histogram of the ADC bins loading plotted

in Figure 7.20, one can see, that for the ISR for loss of lock, which was determined to be 32.5 dB, the signal was only present in 5.5 *bit* of the total 6 *bit* in that ADC.

7.5.2. Evaluation of the Impact of WB-RFI on AGC/ADC

Figure 7.17 showed, that for high power WB-RFI saturation effects in the ADC occur. Hence the WB signal with the parameters presented in Section 7.2.3 was driven to an AGC to mitigate saturation effects in the ADC. The circuit used here is presented in Figure 3.8. Figure 7.22 shows the change in gain versus time on the left and on the right the histogram of the ADC bins distribution versus WB-ISR, for the connection between time in the WB-file and its corresponding ISR see Figure 7.5.

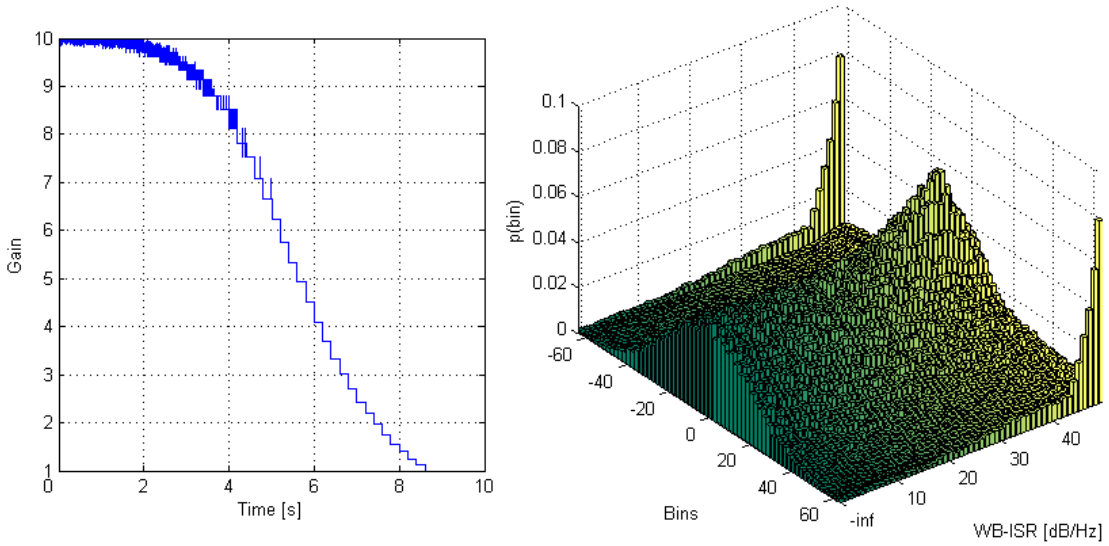

Figure 7.22.: Gain adaption and ADC bins loading in a 6 *bit* ADC for an increasing WB-ISR.

As one can see, due to the gain reduction saturation effects in the ADC are decreased, but because of the limited AGC range saturation effects occur for WB-ISRs $>$ 42 dB. The not Gaussian shape of the ADC bins distribution in the AGC range is a consequence of the error in signal generation of WB interference in the DLR Software Signal Generator, unfortunately the source of that error could not be found to the time of proposal.

After the signal was quantized, the quality of the received GPS signal was assessed in the DLR-Software Receiver. Figure 7.23 shows the obtained results.

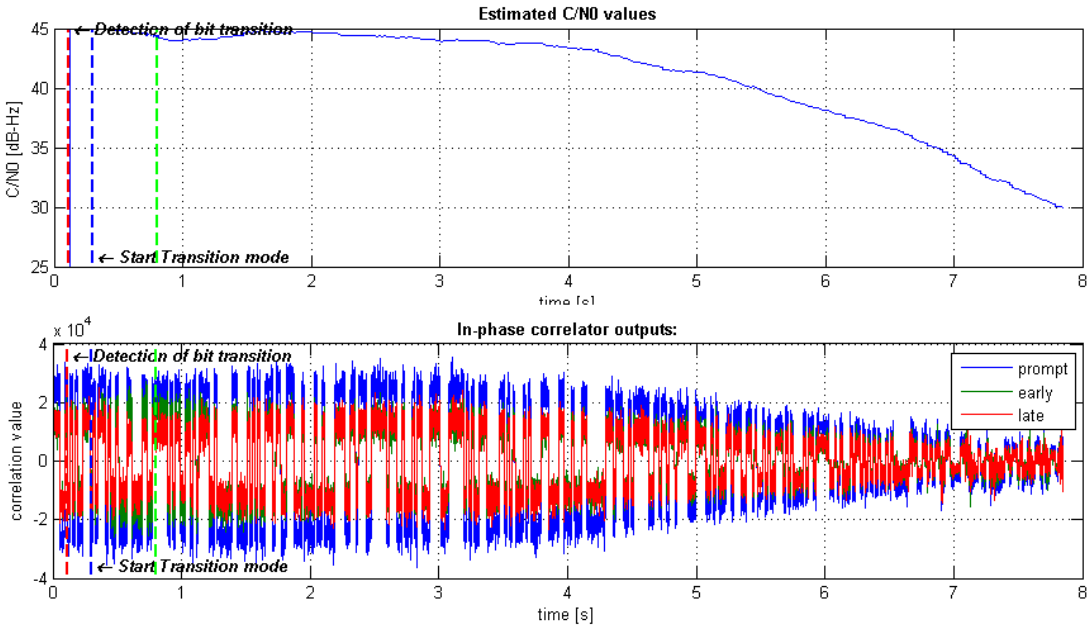

Figure 7.23.: Above: C/N_0 at the receiver output; Middle: Correlator output; Below: Estimated PRN code start; all for a 6 *bit* ADC under the impact of WB-RFI power increasing

Comparing these results with those for the no AGC case, as shown in Figure 7.2.3, attenuating the GPS signal superposed with WB-RFI and noise, leads to a earlier loss of lock and worses the signal's quality at the correlator output.

But however, the loss of lock occurred after 7.8 s of tracking which equates a WB-ISR of about $\approx 37\ dB/Hz$, which is about 17 dB/Hz for a 4 MHz bandwidth above the noise floor. For such high WB interference powers, which are lather unlikely, the ability of the C/A code to reject the impact of WB interference on the signal is limited.

7.6. Simulations on Frequency Excision Techniques with a 6-Bit ADC

This Section focuses on the mitigation of CW-RFI using those frequency excision techniques presented in Section 6.3 in combination with a 6 *bit* ADC. This Section does not include simulations on frequency excision techniques dealing with pulsed interference, because the complete ADC range was initially used, i.e. there is no dynamic range, which is necessary for a proper detection and mitigation of pulsed RFI in the frequency domain.

For the performance investigations of the frequency excision techniques the circuit design presented in Figure 6.3 was used, with the small modification that instead of that 14 *bit* ADC only a simple 6 *bit* ADC was implemented.

7.6.1. Performance Evaluation of Frequency Excision Techniques dealing with CW-RFI

Due to the fact that the circuit's input file, which is depicted in Section 7.2.1, was also used in context with performance evaluation of the GPS C/A code, some results are already presented. Such as the histogram, showing the ADC bins loading versus time depicted in Figure 7.13.

On the basis of the ADC output data the FFT was computed, whereby each data vector had a temporal length of 1 ms, containing $2 \cdot 10^4$ samples and a rectangular data weighting window was used. After that the interference detection was performed in the magnitude spectrum, Figure 7.24 shows the percentage of detected discrete frequencies inside the filter bandwidth which exceeded the detection threshold of $T_{freq}(p_{fa} = 1\%)$.

As one can see, the false detection rate at the beginning, where noise only is present, is about 1.6%, which is greater than the pre-defined one of $p_{fa} = 1\%$. This is due to the fact that for the detection threshold computation a flat rectangular bandpass filter was assumed, the energy loss to the edges of the filter was not took into considerations, see Section 5.5 for a discussion of the threshold determination.

In the continuous analog Fourier spectrum, that percentage of threshold violations would have stayed constant around 1.6%, because there is only a Dirac pulse at the frequency of the CW-RFI, else there are only variations due to noise. But here, as shown in Figure 7.24 the false detection rate decreases after 6 s of processing, which is

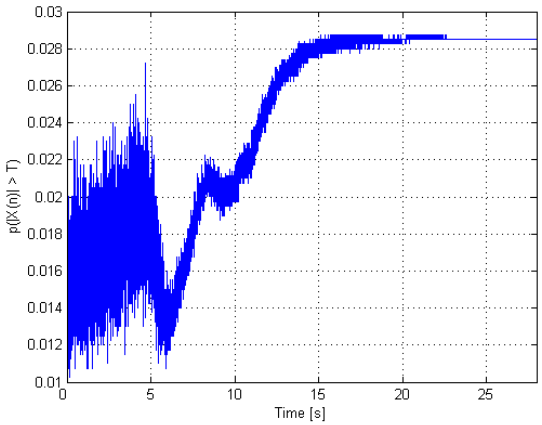

Figure 7.24.: Percentage of detected discrete frequencies $> T_{freq}(p_{fa} = 1\%)$ for a 6 *bit* ADC, using rectangular data windows.

qual to a CW-ISR of 27 dB, because when observing the historgam of the ADC bins loading versus CW-ISR, presented in Figure 7.13, saturation effects for $ISR > 27\ dB$ occur in the ADC. The increase of the percentage $p(|\mathbf{X}(n)| > T_{freq}(p_{fa} = 1\%))$ after 7 s has the same reasons, which are the saturation and non linear effects in the ADC. These effects also lead to the circumstansce that the false detection rate stays constant at 2.85 % after 20 s of processing.

After the frequency excision techniques were performed and the signal was software quantized so that the output samples have a 6 *bit* resolution, the now interference free signal was investigated in the DLR Software Receiver, the results are shown in Figure 7.25. Recap, that software quantization is necessary because the output of the IFFT delivers complex values of type “double”, but for further signal processing the samples shall be real and of type “integer” with a 6 *bit* resolution.

Comparing the results with those where the C/A code had to deal with the increasing CW-ISR on its own (see Figure 7.14), the receivers performance was drastically increased. In the earlier example the receiver was only able to deal with a CW-ISR of 29.5 dB, but due to the frequency excision techniques it is now able to track up to CW-ISR of $\approx 50\ dB$, i.e. the receivers performance was increased by 20 dB for dealing with CW-RFI sources.

Also interesting to mention is that for such a CW-ISR level of 50 dB in combination with the frequency excision techniques, the signal needs only to spend 4.65 % of the time between the maximum ADC quantization thresholds to keep lock. Figure 7.26 illustrates the bins loading for a CW-RFI under these conditions. When comparing this figure with the case for the C/A code dealing alone with ADC saturation effects

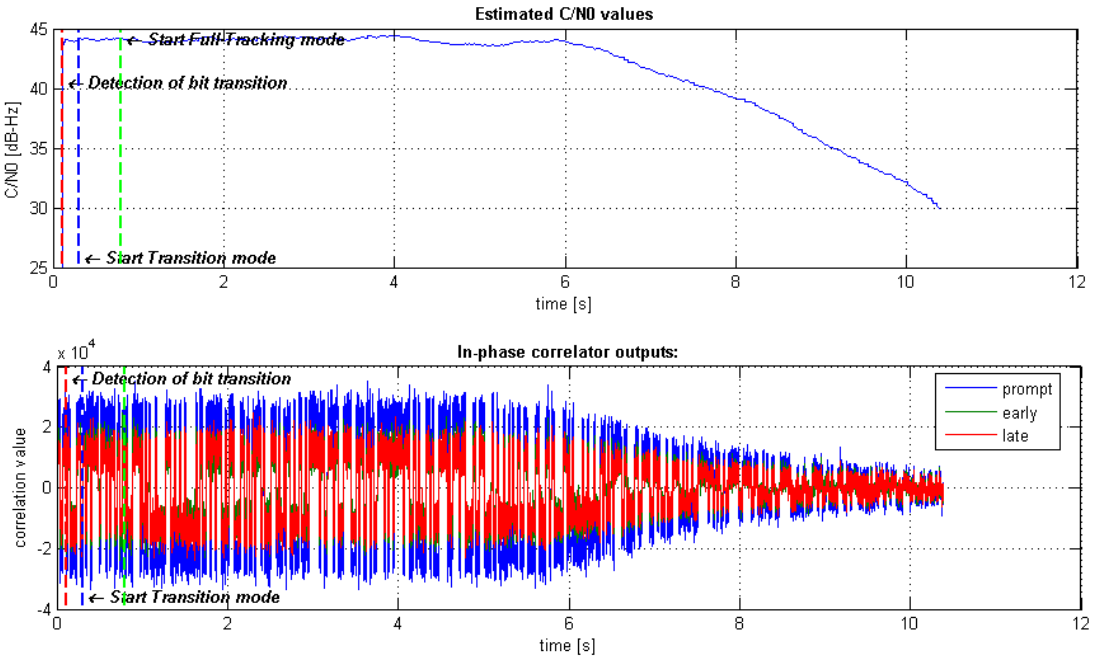

Figure 7.25.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 6 *bit* ADC in combination with frequency excision techniques, using rectangular data windows, under the impact of an increasing CW-RFI power.

and CW-RFI, presented in Figure 7.15 the enormous performance of the frequency excision techniques is clarified, because there the signal needed to spend 58.25 % of the time between the maximum ADC bins to keep lock.

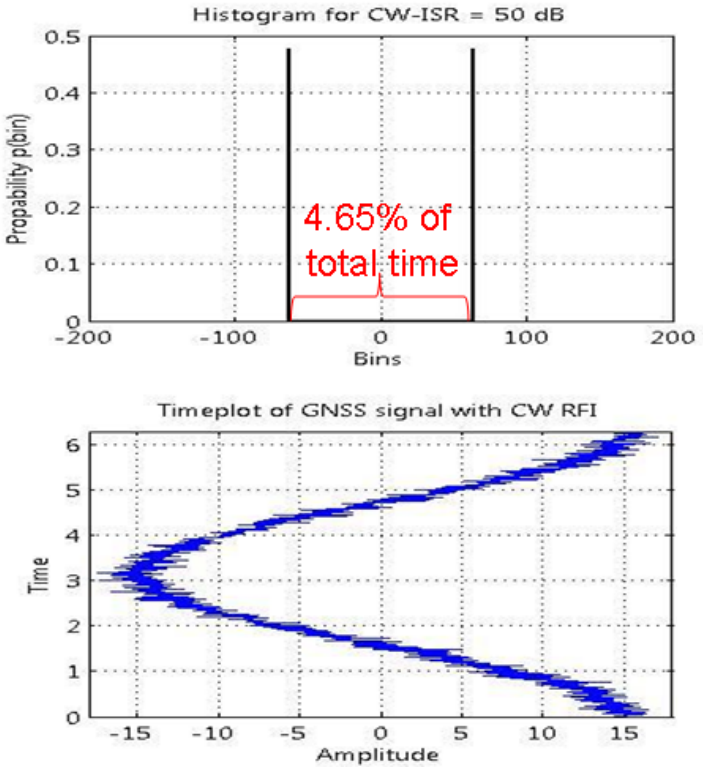

Figure 7.26.: Time in % that the signal spended between the maximum ADC bins to keep lock for a $CW - ISR = 50\ dB$, in combination with frequency excisoin techniques.

7.7. Simulations on Frequency Excision Techniques with a 6-Bit AGC/ADC

7.7.1. Performance Evaluation of Frequency Excision Techniques dealing with CW-RFI

Last Section provided the simulations on a 6 *bit* ADC with subsequent frequency excision techniques using rectangular data windowing. As shown, the frequency excision techniques have a enormous performance to reject CW-RFI even if its power is quite large and saturation effects in the ADC occur. But however, the performance of the frequency excision techniques dealing with strong CW-RFI and saturation effects is limited, which make it necessary to implement an AGC. Therefore the circuit presented in Figure 6.4 was used to reduce saturation effects in the ADC. But notice here the ADC has in contrast to the one illustrated in Figure 6.4 only 6 *bit*, and further the pulse detection circuit shown in Figure 6.4 was not used, because only CW-RFI is concerned.

The typical gain adaption curve and ADC bins loading plot is the same as already presented in Section 7.5.1, because the same input source with an increasing CW-ISR was used, see Section 7.2.1 for description.

But before proceeding with the performance evaluation of the frequency excision techniques, the impact of the AGC on the FFT computation needs to be examined first. Therefore, Figure 7.27 shows the computed discrete spectrum and the adaptive threshold $T_{freq}(p_{fa} = 1\%)$ for the maximum AGC amplification of 20 dB on the left and the minimum AGC amplification of 0 dB on the right. The total available gain g_0 was set according to ratio k_{opt} between the maximum quantization threshold and the input standard deviation for the RFI free case ($g_0 = g_{front} + max(AGC)$).

Obviously for the RFI free case the SNR is about 17 dB, but due to the gain adaption the SNR is decreased by 10 dB, because for an AGC setting of 20 dB the signal was quantized with 6 *bit*, but for an AGC setting of 0 dB the signal is only quantized with about 2.5 *bit*. Thus a serious amount of quantization noise is introduced to the signal, which also leads to an increase of the false detection rate to 0.2 %.
Notice: Considering the left plot again, when an AGC shall be implemented the designer has to regard the initial SNR, because when the AGC would be able to attenuate the incoming signal beyond 20 dB, one would finally come to the point of hardlimiting. But for this case the linear threshold adaption strategy is no more valid and problems in the discrete spectrum estimation will occur due to the low signal resolution. Hence the gain should not be decreased more than $\approx$ 30 dB for

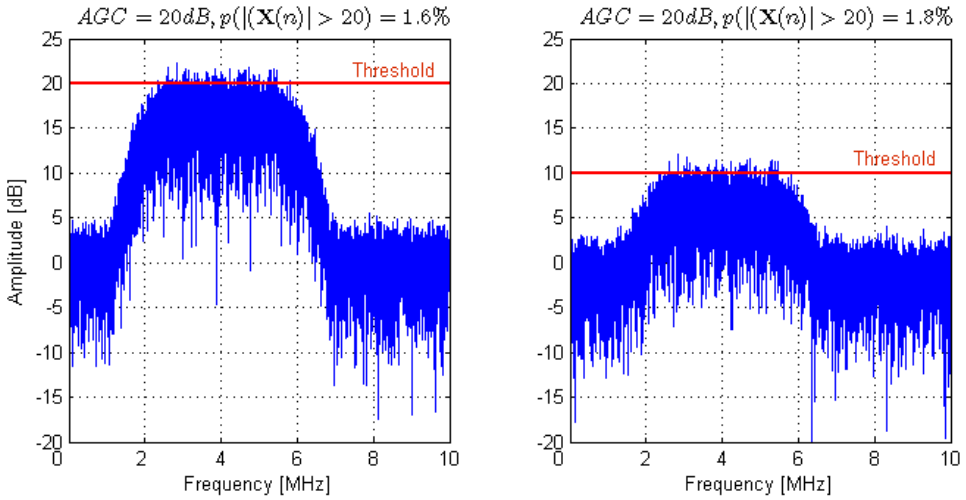

Figure 7.27.: Impact of the AGC on the FFT computation; Left: 20 dB AGC amplification; Right: 0 dB AGC amplification

this case.

Next the performance of the frequency excision techniques was assessed, similar to the discussions in the last section, Figure 7.28 shows the percentage of the discrete spectral lines exceeding the spectral detection threshold versus time. Like last time the FFT was computed of 1 ms long vectors, containing $2 \cdot 10^4$ quantized samples, using a rectangular window function. The spectral interference detection threshold was defined according to a false detection rate of $p_{false} = 1\%$.

Although the saturation effects in the ADC were reduced, see Figure 7.20 for the ADC bins loading, the percentage of detected discrete frequencies exceeding the threshold $T_{freq}(p_{fa} = 1\%)$ increased even stronger with increasing CW-ISR now, compared with the no AGC case. This is due to the fact that earlier the nonlinear saturation effects in the ADC limited the leackage effect.

But here, due to the gain adaption, saturation effects are reduced, hence the signal is quantized "better" for higher CW-ISRs as for the no AGC case. Because of the "better" A/D-conversion one might expect the FFT to do a better approximation of the analog spectrum, too, but because of the circumstance that the analog CW-RFI frequency is exactly halfway between two discrete spectral lines, strong leakage effects in the discrete spectrum computation occur.

But anyway, the data of the circuit's output was investigated in the Software Receiver, the results are plotted in Figure 7.29.

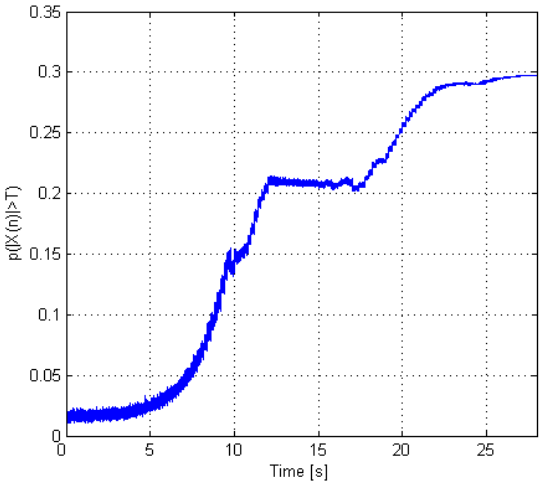

Figure 7.28.: Percentage of detected discrete frequencies $> T_{freq}(p_{fa} = 1\%)$ for a 6 *bit* AGC/ADC, using rectangular data windows.

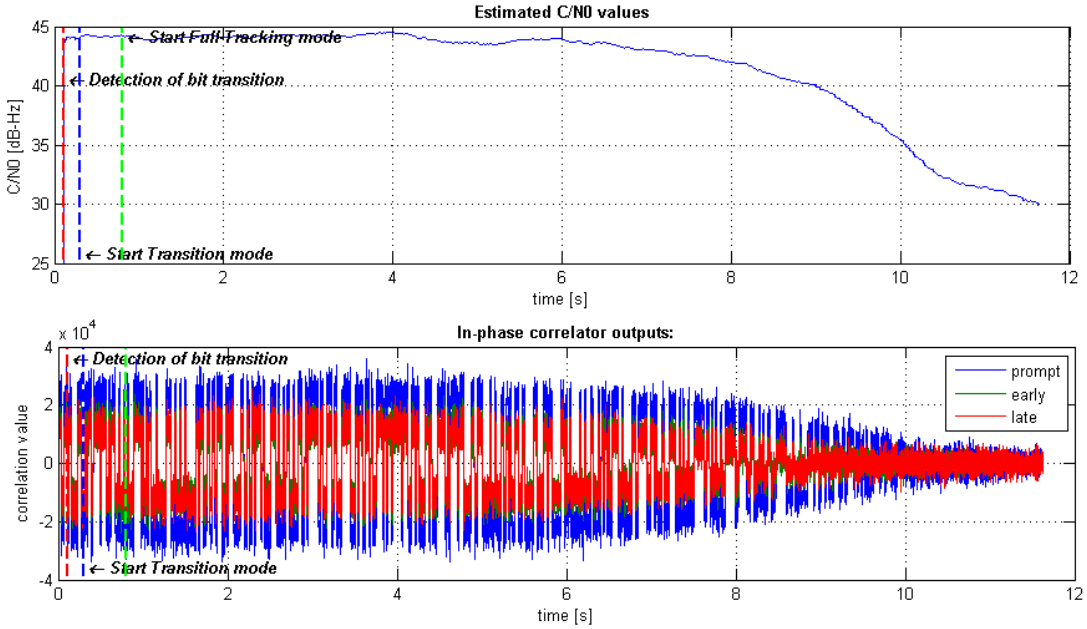

Figure 7.29.: Above: C/N_0 at the receiver output; Below: Correlator output; all for an 6 *bit* AGC/ADC in combination with frequency excision techniques, using rectangular data windows, under the impact of an increasing CW-RFI power.

As one can see the achieved C/N_0 and the correlator output signal quality is slightly better as for the no AGC case, but anyway the receiver lost lock at approximately the same time and so the same CW-ISR which was $\approx 50\ dB$, like in the last section.

In order to decrease the spectral leakage and so increase the receiver's performance dealing with strong CW-RFI sources, the Hanning window was proposed in Section 5.5. Its effect on the computation of the discrete spectrum using the FFT is shown Figure 5.18, where the 1 ms long data vector, containing $2 \cdot 10^4$ "analog" signal samples was used to compute the discrete signal spectrum of the GPS signal superposed with noise and a CW-RFI with an ISR of 60 dB.

The next Figure 7.30 shows the percentage of discrete frequencies exceeding the spectral detection threshold of $T_{freq}(p_{fa} = 1\%)$ versus time, for the circuit's input the file containing a CW-RFI with an increasing ISR was chosen, see Section 7.2.1 for file description.

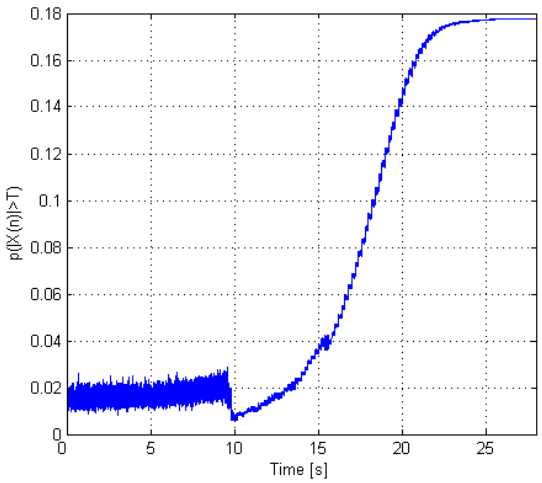

Figure 7.30.: Percentage of detected discrete frequencies $> T_{freq}(p_{fa} = 1\%)$ for a 6 bit AGC/ADC, using Hanning data windows.

Of course, the detection rate should have stayed constant around the pre-defined false alarm rate of 1 % (here 1.6 % due to the no consideration of energy loss at the filter edges), as long as spectral leakage can be neglected for the Hanning window. But due to the limited AGC range saturation effects in the ADC occur for CW-ISRs $> 48\ dB$, which are present in that input file for times $> 10\ s$, due to which the detection rate $p(|\mathbf{X}(n)| > T_{freq}(p_{fa} = 1\%))$ increases.

After the interference was rejected with the frequency excision techniques using Hanning data windows, the obtained signal was driven to the DLR Software Receiver,

the results are shown in Figure 7.31

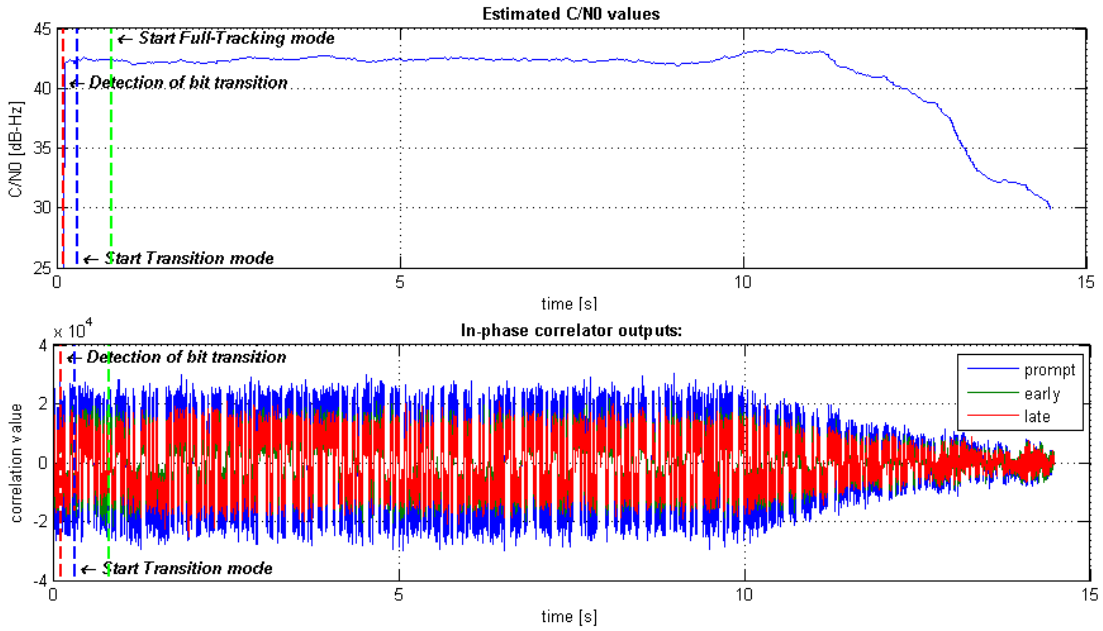

Figure 7.31.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 6 *bit* AGC/ADC in combination with frequency excision techniques, using a Hanning windowing, under the impact of CW-RFI power increasing

Obviously, the receiver's robustness was improved. Although the C/N_0 fell here below the loss-of-lock threshold after the $14st$ second of processing which is equal to an ISR of 68 dB, the real loss of lock is earlier. It was estimated to be at about 62 dB, for that ISR the percentage of discrete frequency detection exceeding that threshold is still about 2 % but due to the strong ADC saturation effects the signal quality is decreased that much that no tracking is anymore possible.

Last but not least, also the over all C/N_0 and correlator output energy degradation needs to be concerned, as they are the consequence of temporal windowing. As already explained in Section 5.5, every window process leads to a loss of signal energy, for the Hanning window this loss is about 6 dB on the signal but ≈ 1.42 dB at the corrector output. In order to avoid this energy loss, the concept of overlapped Hanning windows was introduced in Section 5.5.5.

For the following simulation, the circuit depicted in Figure 6.6 was used, in combination with overlapped Hanning windows. The figures of the gain adaption curve, ADC bins loading and the percentage of discrete frequencies exceeding the spectral threshold are the same as above. Only the achieved receiver performance is different. In order to clarify this Figure 7.32 shows obtained receiver results.

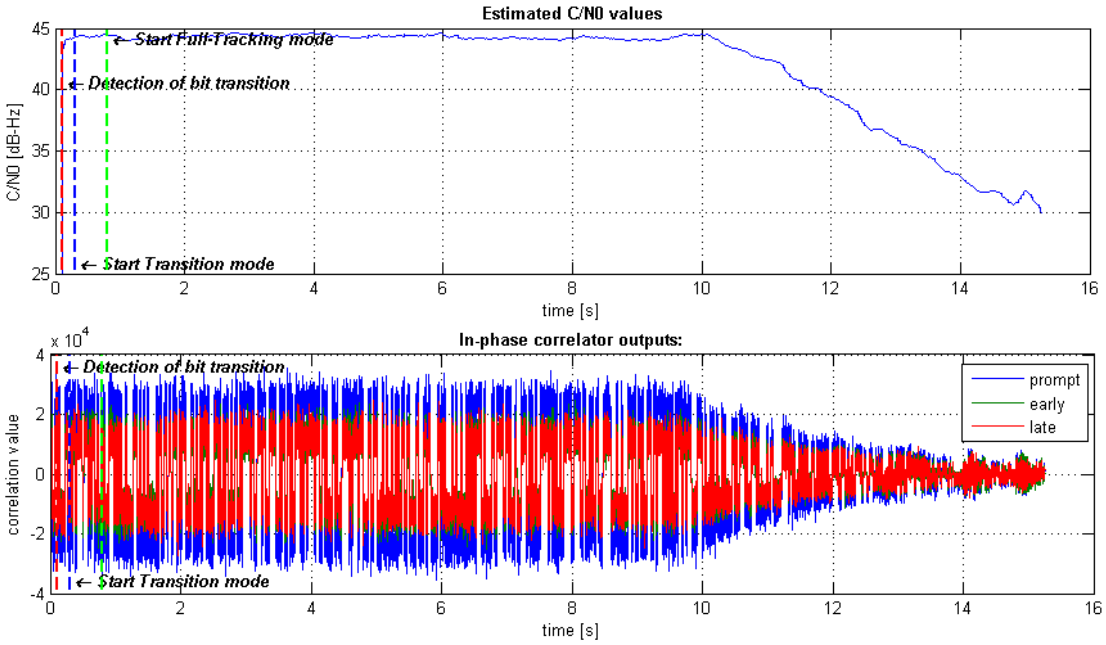

Figure 7.32.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 6 *bit* AGC/ADC in combination with frequency excision techniques, using overlapped Hanning windows, under the impact of CW-RFI power increasing

It seems that due to the concept of overlapped Hanning windows no C/N_0 degradation occurs, because those samples that are fully attenuated by one window, are applied with the full gain in the next window and spectral leakage is reduced due to the better spectral characteristic of the Hanning window compared with the rectangular one.

This section assessed the performance of a circuit with a 6 *bit* AGC/ADC in combination with frequency excision techniques when using different window types to reduce spectral leakage in the FFT computation. It was shown that overlapped Hanning windows supply the best performance for the rejection of CW interference when a 6 *bit* AGC/ADC circuit is used for the transformation to the digital state, although some issues arise due to the implemented gain adaption strategy, which was known not to be the best, because it is only optimal for Gaussian thermal noise. Then as already shown in the histogram of Figure 7.20, for an increasing CW-RFI power the signal is temporary compressed into ≈ 45 of the total available 64 ADC bins due to the gain reduction. So, a better signal quality could be achieved when the signal is always spread over all available ADC bins. Unfortunately for such a gain adaption strategy a priori knowledge of the occurring interference type and its PDF is necessary, which was not available. But anyway, the frequency excision techniques did such a fine job that this issue could not be observed in a degradation

of the achieved C/N_0 nor the correlator output signal quality.

In the following discussions a 14 *bit* ADC with a 6 *bit* initial loading was used, so that the problem of AGC steering could be diminished by modifying the AGC's task to avoid only saturation effects in the ADC.

7.8. Simulations on Frequency Excision Techniques with a 14-Bit ADC

This section investigates the circuit presented in Figure 6.3, where the frequency excision techniques are used in combination with a 14 *bit* ADC, having more bits than required for further signal processing. Hence the total gain g_0 was set according to a 6 *bit* initial ADC loading. Figure 7.33 shows the ADC bin loading for only Gaussian noise present.

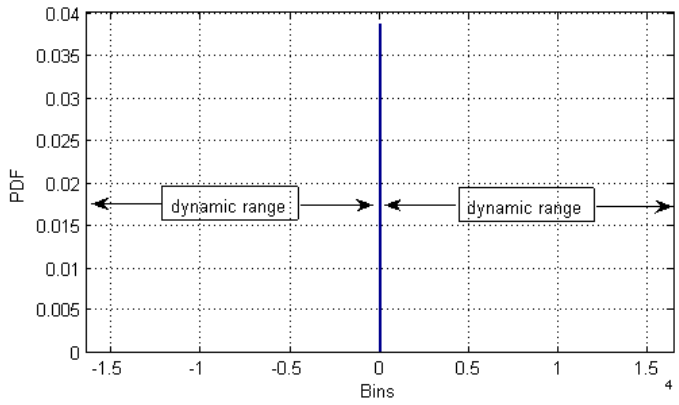

Figure 7.33.: Initial 14 *bit* ADC loading.

One can see, the ADC has under these condition an enormous dynamic range. The nominal ADC loading of noise is compressed into a "line", because the signal is quantized into only 64 bins of the total 16384 bins. But although not visible here, the loading of these lowest 64 bins clearly represent the Gaussian distribution of noise, as illustrated in Figure 7.6.

When now interference occurs, the signal is spread over the remaining ADC bins due to the increased signal input power.

This section presents the simulation results of that circuit dealing with CW and pulsed-RFI. Thereby the FFT is computed of 1 ms long vectors, containing $2 \cdot 10^4$ samples, like always in this work.

7.8.1. Performance Evaluation of Frequency Excision Techniques dealing with CW-RFI

At first, the impact of an increasing CW-ISR on the performance of the frequency excision techniques was investigated, taking the CW file with the parameters presented in Section 7.2.1 as input signal.

Similar to the discussions in Section 7.6, a rectangular data windows were used for the FFT computations. Next the interference detection in the spectrum was performed, Figure 7.34 shows the percentage of the discrete spectral lines exceeding the spectral detection threshold defined according to a false detection rate of 1 % versus time.

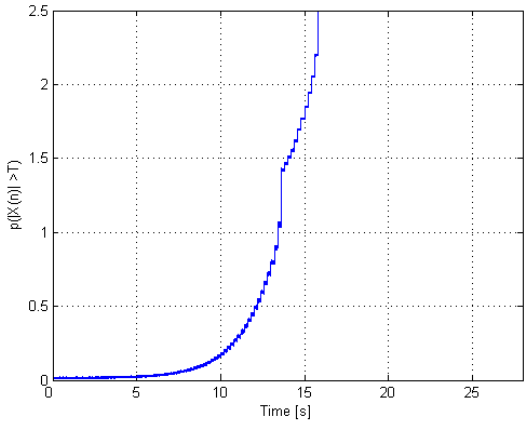

Figure 7.34.: Percentage of detected discrete frequencies $> T_{freq}(p_{fa} = 1\%)$ for a 14 *bit* ADC, usings rectangular data windows.

As one can see, the percentage of detected threshold violations increases with increasing CW-powers. At the time of ≈ 13 s, which is equivalent to a CW-ISR of 63 dB, that percentage exceeds 100 %, meaning that for an ISR > 63 dB the energy of the CW-RFI, leaks even into frequencies outside the filter bandwidth (the detection rate was normalized according to the front-end filter bandwidth). This increase has nothing to do with the limited ADC range, because its borders are not reached for a CW-ISR of 63 dB. In order to clarify this, Figure 7.35 shows the ADC bins

loading on the left and on the right the computed discrete spectrum using the FFT with a rectangular data window function.

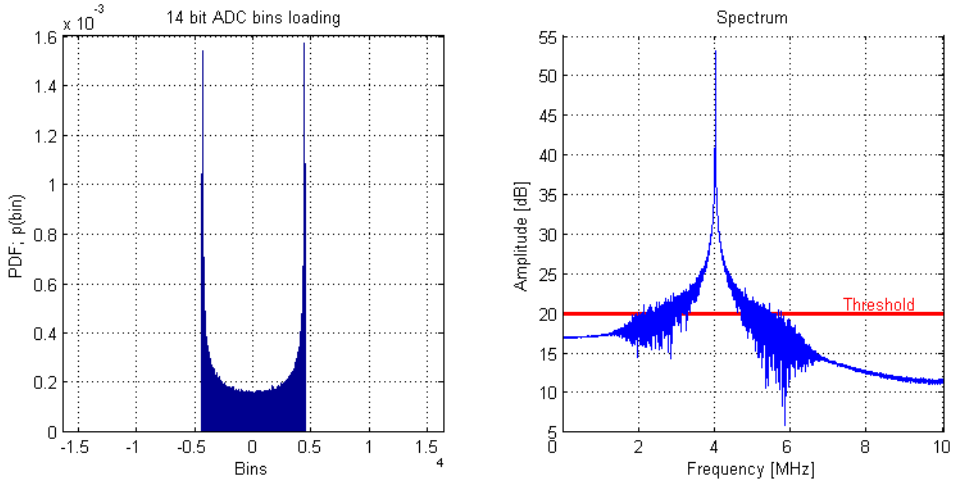

Figure 7.35.: CW-ISR of 63 *db*; Left: 14 *bit* ADC bins loading; Right: Computed discrete spectrum using a rectangular window function

As one can see, the incoming signal is only present in a small range of the ADC for a CW-RFI with an ISR of 63 dB, but as illustrated on the right, the leakage effect corrupts the discrete spectrum estimation for such high CW powers.

Although it was shown that the frequency excision techniques did not work properly for high CW powers, the circuit's output signal was provided to the DLR Software Receiver to investigate the achieved signal quality, Figure 7.36 shows the results.

The receiver lost lock after ≈ 10.6 s of processing, but focusing on the correlator output, the signal quality was degraded that much that already after 9.4 s, which is equivalent to a CW-ISR of 50 dB, no real signal processing can take place. The here obtained performance is equal to that of the simple 6 *bit* AGC/ADC circuit in combination with the frequency excision techniques using rectangular windows.

Hence, overlapped Hanning windows were used in the following to reduce spectral leakage in the discrete spectrum computation and so to obtain a better analog spectrum approximation. Figure 7.37 shows the computed spectrum for a CW-ISR of 63 *db* on the basis of the quantized data and a Hanning data window function, the energy loss due to windowing was concerned before displaying. Comparing that spectrum with the one where a rectangular data window function was used, shown in Figure 7.35, this one clearly represents the spectrum of filtered noise superposed with a CW-RFI.

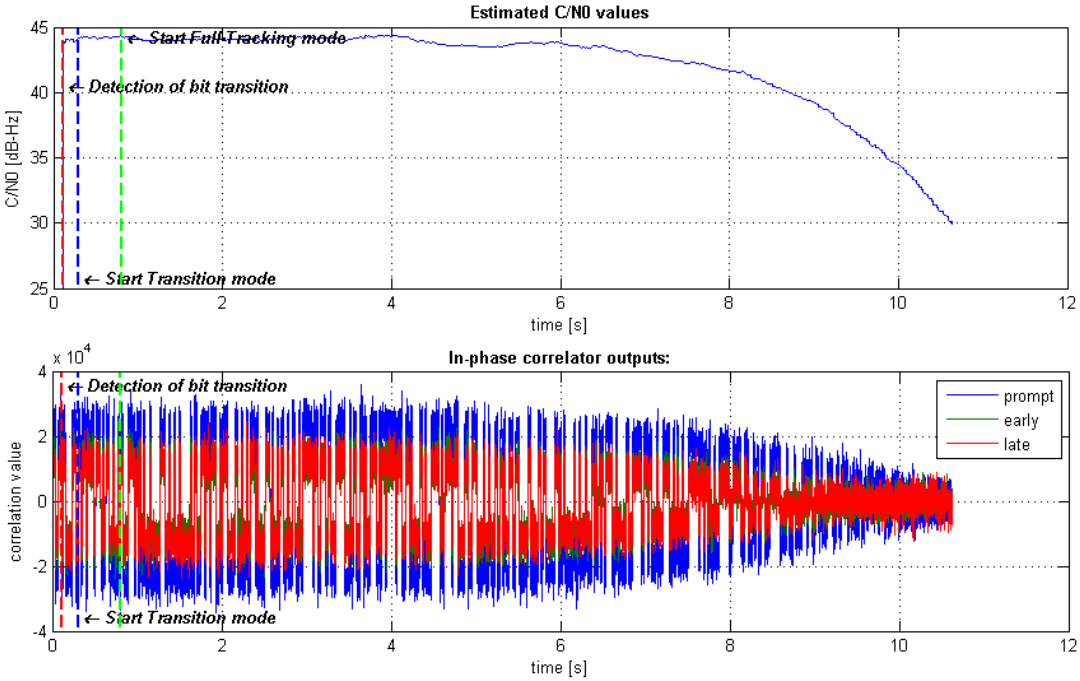

Figure 7.36.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 14 *bit* ADC in combination with frequency excision techniques, using rectangular data windows, under the impact of an increasing CW-RFI power

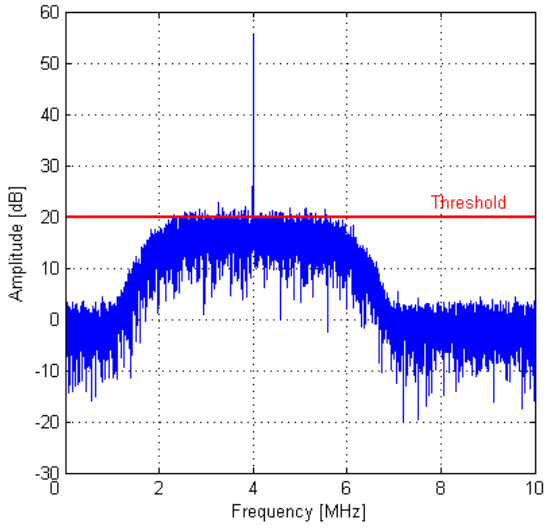

Figure 7.37.: Computed discrete spectrum using a Hanning window function, for a $CW - ISR = 63\ dB$

Of course, with the help of Hanning data windows the leakage effect can even be neglected beyond an ISR level of 63 dB, in the case of a 14 bit ADC the leakage effect does not occur in mentionable manner, before the ADC is in saturation. Therefore Figure 7.38 shows the percentage of detected discrete frequencies exceeding the spectral detection threshold versus time.

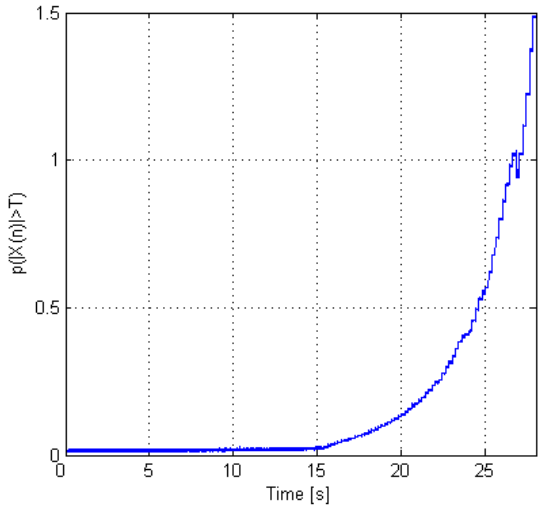

Figure 7.38.: Percentage of detected discrete frequencies $> T_{freq}(p_{fa} = 1\%)$ for a 14 bit ADC, Hanning window function.

Obviously, it stays fairly constant up to the $15st$ second of processing, which is equivalent to an ISR of 73 dB. After that it increases due to saturation effects in the ADC. The quality of the obtained data was investigated in the DLR Software Receiver. The results are plotted in Figure 7.39

As one can see, the receiver is able to track up to the $19st$ second of processing. But considering the correlator output signal quality which is degraded after the $17st$ second of processing, i.e. for a CW-ISR of 83 dB that much, that no real signal processing is anymore possible.

It is remarkable, that for this CW-ISR, the signal spend only 25 % of its total time between the two highes ADC bins and further the frequency excision techniques excluded about $\approx$ 10 % of the signals bandwidth, but the receiver was still able to proceed tracking.

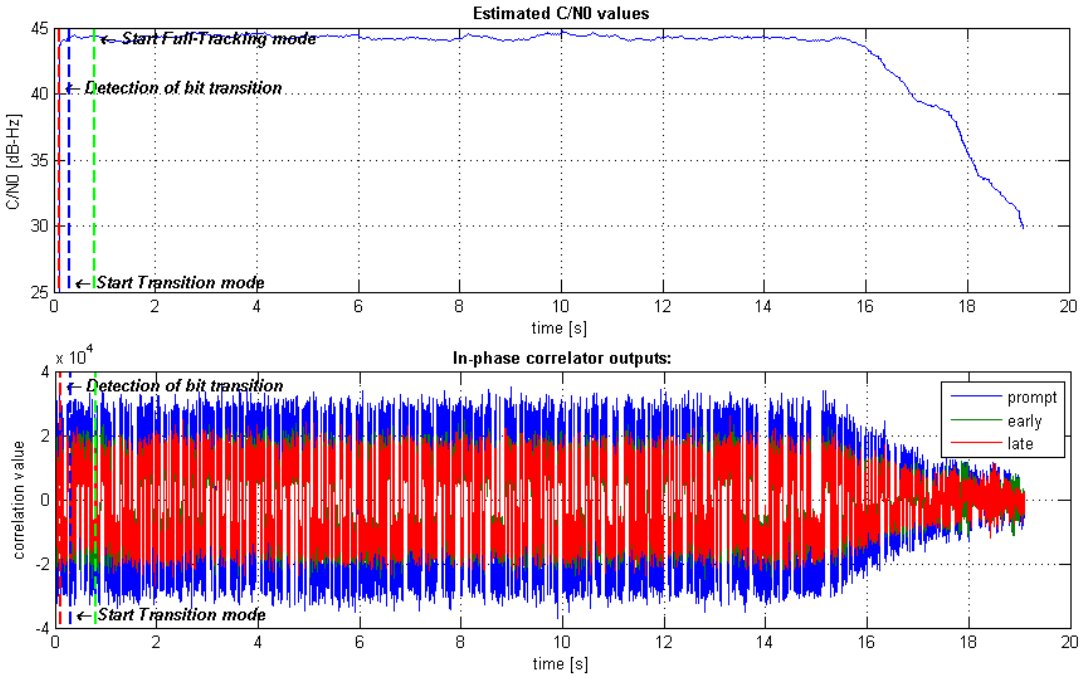

Figure 7.39.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 14 *bit* ADC in combination with frequency excision techniques, using overlapped Hanning windows, under the impact of an increasing CW-RFI power

7.8.2. Performance Evaluation of Frequency Excision Techniques dealing with Pulsed-RFI

For the performance evaluation of the frequency excision techniques dealing with pulsed RFI, the circuit input file presented in Section 7.2.1 was used. The quantized output of a DME plus pair is depicted in Figure 7.40.

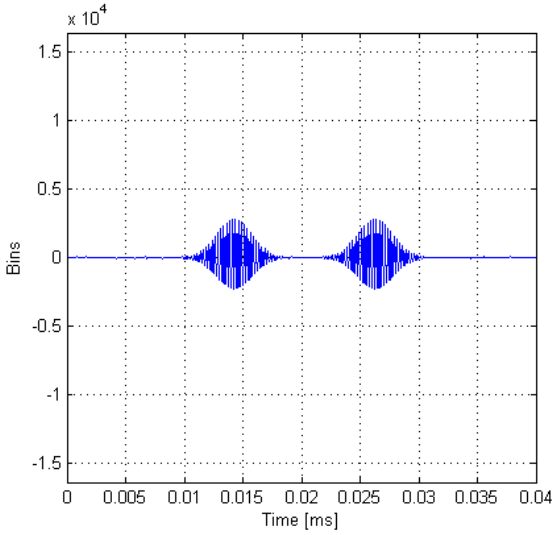

Figure 7.40.: Time plot of quantized DME data of a 14 *bit* ADC.

As one can see, the ADC loading is not even near its borders for a DME interference with a PSR of 60 *db*.

On the basis of the quantized data the FFT was computed. Figure 7.41 shows the obtained discrete spectrum using a rectangular and a Hanning data window function.

Obviously, the discrete spectrum using the FFT in combination with a rectangular data window suffers strongly under the leakage effect. Whereas the discrete spectrum obtained when using a Hanning window seems to provide a good approximation of the continuous Fourier spectrum, so that the frequency excision techniques should do a better job for this data window function.

In Section 7.4.2 it was shown that the receiver does not lose lock in the case that the pulsed interference is quantized with a 6 *bit* ADC, only the C/N_0 is decreased by 7 *dB*. Hence the frequency excision techniques were performed on the signal to reduce the C/N_0 loss. The obtained output file was driven to the DLR Software Receiver, the results are plotted Figure 7.42, where the C/N_0 is shown for the case

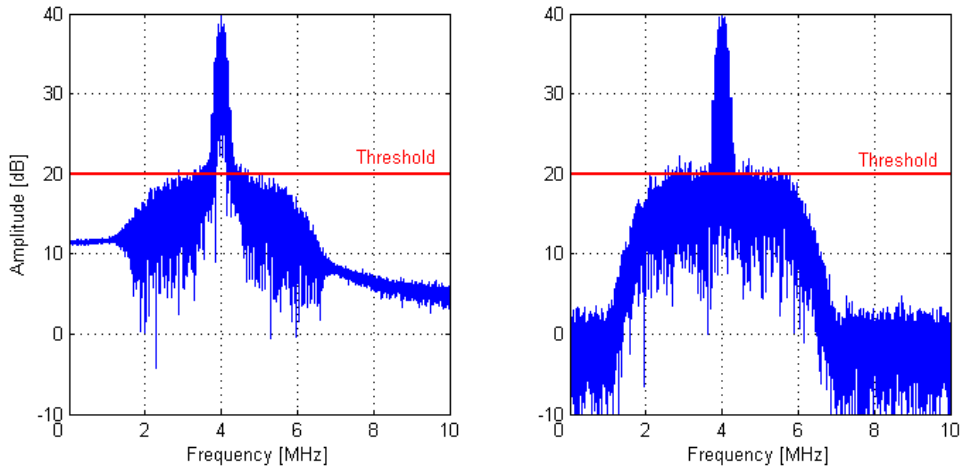

Figure 7.41.: Magnitude spectrum of quantized signal for a DME-PSR of 60 dB, computed with the FFT, using a rectangular window on the left and a Hanning window on the right.

that the frequency excision techniques were used in combination with rectangular and Hanning data windowing to compute the FFT.

Comparing the achieved C/N_0 levels with those obtained for a 6 *bit* ADC where the C/A code mitigated the pulsed interference on its one, shown in Figure 7.16, it catches ones eyes that using a 14 *bit* ADC in combination with frequency excision techniques and rectangular data windowing, the GPS signal quality is degraded even more, but tracking is still possible. This increased C/N_0 degradation is the direct results performing the frequency excision techniques on a spectrum that suffers strongly under the leakage effect. But when using overlapped Hanning windows, the frequency excision techniques can work properly and reject the interference better. But anyway the C/N_0 suffers a degradation of ≈ 3 dB compared to the noise only case.

7.9. Simulations on Frequency Excision Techniques with a 14-Bit AGC/ADC

In Section 7.8 the frequency excision circuit illustrated in Figure 6.3 was investigated. It was shown, that the performance of the frequency excision techniques is limited when the 14 *bit* ADC is driven into saturation due to high RFI powers.

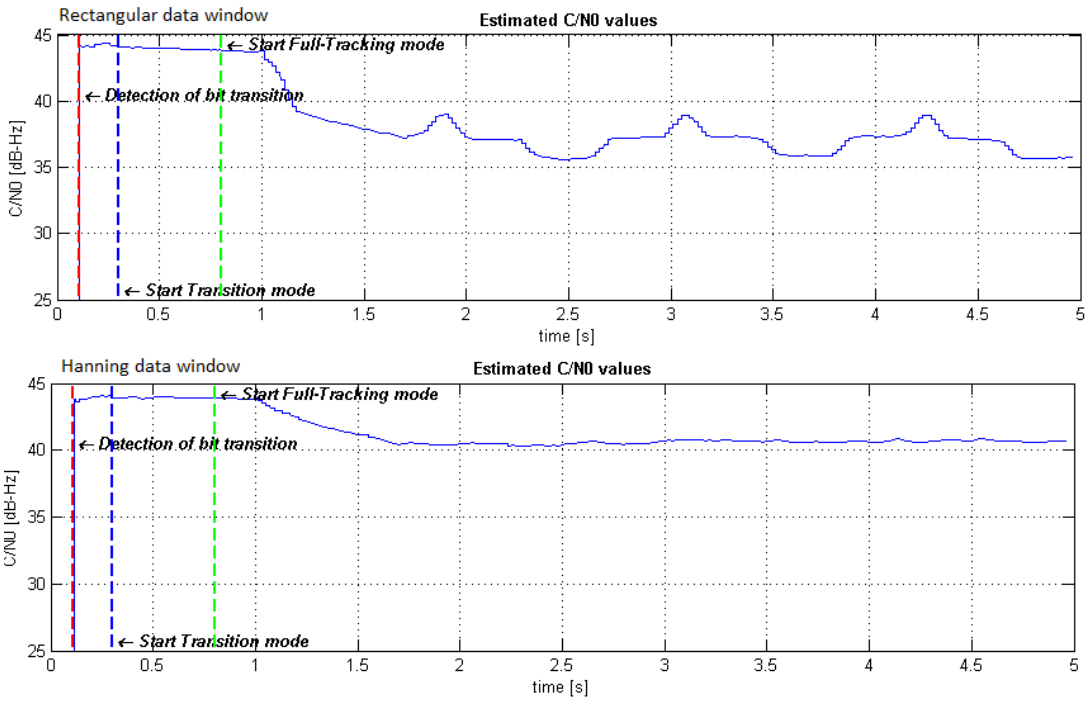

Figure 7.42.: C/N_0 degradation when using rectangular and Hanning data windows to mitigate the impact of pulsed interference in the frequency domain

Hence an AGC was combined with the 14 *bit* ADC to reduce saturation effects. This section examined the performance of the circuit presented in Figure 6.6, where a 14 *bit* AGC/ADC is combined with the frequency excision techniques. Pulsed interference is not investigated in this section, due to the fact that they are not allowed to affect the AGC, and so the results are equal to those shown in Section 7.8.2, assuming that no other form of interference and no fluctuations in the received noise floor occur. Another parameter jet to discuss is the implemented data window function. The simulations in Section 7.8 showed that for a CW-ISR $> 50\ dB$, the 14 *bit* ADC with a 6 bit initial loading is not in saturation, but the leakage effect worse the approximation of the continuous analog signal spectrum by the discrete spectrum that much, that the frequency excision techniques could not work anymore properly. Therefore the concept of overlapped Hanning windows was introduced, to reduce spectral leakage in the FFT computation and to avoid energy loss due to windowing. All simulations presented in this section use the concept of overlapped Hanning windows for the FFT computation.

7.9.1. Performance Evaluation of Frequency Excision Techniques dealing with CW-RFI

Similar to the previous discussions on the mitigation of CW-RFI, the data file depicted in Section 7.2.1 was driven to the AGC/ADC input. On the quantized data, AGC steering and frequency excision techniques were performed. Due to the fact that one has not to deal with pulsed interference here and it is difficult to distinguish between pulsed and CW interference if the CW power becomes large, the pulse detection unit depicted in Figure 6.6 was not used. Another parameter jet to discuss is the total available gain. Recall: The specifics were that the data shall have a resolution of 6 *bit* when noise is the prevalent signal and the AGC has a 20 dB range. Hence the total available gain g_0 was set according to the ratio k_{opt}, between the input standard deviation for the interference free case and the maximum quantization threshold L_{new} (see Equation 3.7) of the $6st$ ADC bit.
Although higher ADC loadings would have been possible to avoid that the AGC can worsen the signal quality to much, a 6 *bit* initial loading was taken, because than ADC saturation effects can be limited most and in Section 7.7 it was investigated that when the gain is set to 0 dB, the signal owns still a ≈ 2.5 *bit* resolution, which is enough for a proper analog spectrum approximation by the discrete one and so the frequency excision techniques and the receiver to work properly.

Next to discuss is the gain steering strategy, as there are two. It can either be to regard the optimum ratio k_{opt}, and so keep the ADC input power constant, or to avoid saturation effects in the ADC. Figure 7.43 shows the gain change versus time for these two different gain adaption strategies, when the input file depicted in Section 7.2.1 is driven to the AGC/ADC input.

The left plot shows the gain adaption strategy that tries to maintain a constant input power. Hence that the gain decreases already at the beginning, even for low CW-ISRs. Fluctuations in the gain curve for $t < 5$ s originate from fluctuations in the received noise floor. These fluctuations are not to observe in the right plot, because here the gain decreases only to avoid saturation in the ADC, so power fluctuations in the received noise floor do not affect the AGC because the interference power is dominating. But notice, the AGC reduces its gain already for a CW-ISR of 60 dB but from Figure 7.35 it is known that for a CW-ISR of about 63 dB the ADC loading is not even near its borders. The developed AGC circuit still tries to maintain the optimum ratio k_{opt}, but that is in this case the ratio between the maximum ADC quantization threshold L and the input standard deviation σ. This is known not to be the best strategy, because it does not respect the PDF of a sine. When respecting the PDF of a sine, the look-up-table needs to be generated with this a priori knowledge. But unfortunately this a priori knowledge about the PDF of the

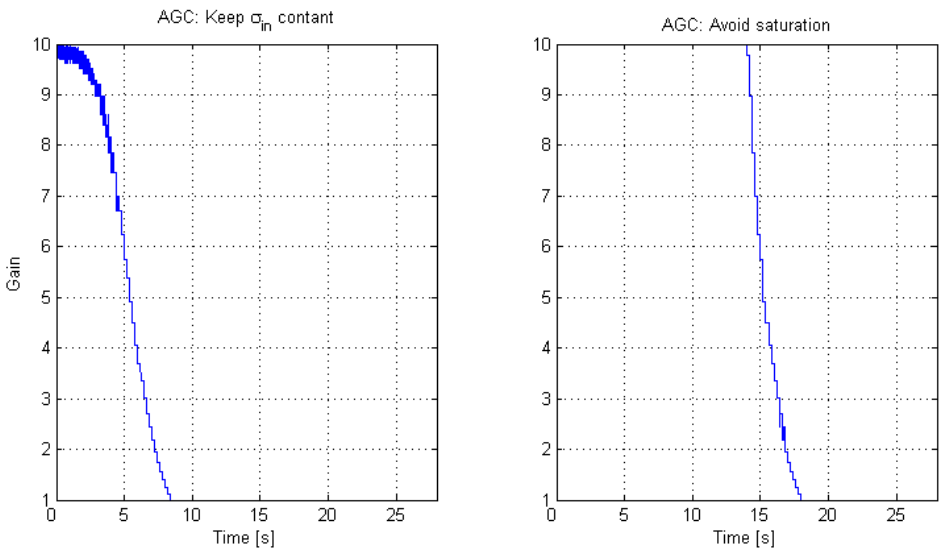

Figure 7.43.: Gain adaption strategy; Left: AGC tries to maintain constant ADC input power; Right: AGC tries to avoid saturation effects

interference is not available in the receiver. Just a gain steering strategy that tried to maintain a Gaussian distribution of the ADC bins could be found in literature and was implemented as discussed in Section 3.3.1.

However, this not optimal gain steering strategy did not affect the developed frequency excision techniques, because the signal is always quantized with a minimum resolution of 2.5 *bit*, which is, as shown in Section 7.7 high enough for the spectrum computation and correct threshold adaption or inverse gain multiplication. Next, on the basis of the quantized data the frequency excision techniques using overlapped Hanning windows were performed, Figure 7.44 shows the percentage of detected discrete frequencies exceeding the spectral detection threshold versus time.

As one can see, the interference detection works fine in both cases. Although the gain in the first plot was in its lower border to an earlier point in time, i.e. for a lower CW-ISR, due to which the resolution of the quantized signal was also decreased earlier, the performance of the frequency excision techniques was not affected. The increase in the detection rate of threshold violations originate from ADC saturation effects which occur for an ISR > 95 dB, spectral leakage can be still neglected for this ISR, because of the good spectral characteristic of the Hanning window.

After the frequency excision techniques were performed the output signal was driven to the DLR Software Receiver, the results are shown in Figure 7.45. Because the plots for both gain adaption strategies were approximately identical, only the results for the gain steering strategy where the avoidance of ADC saturation was objective

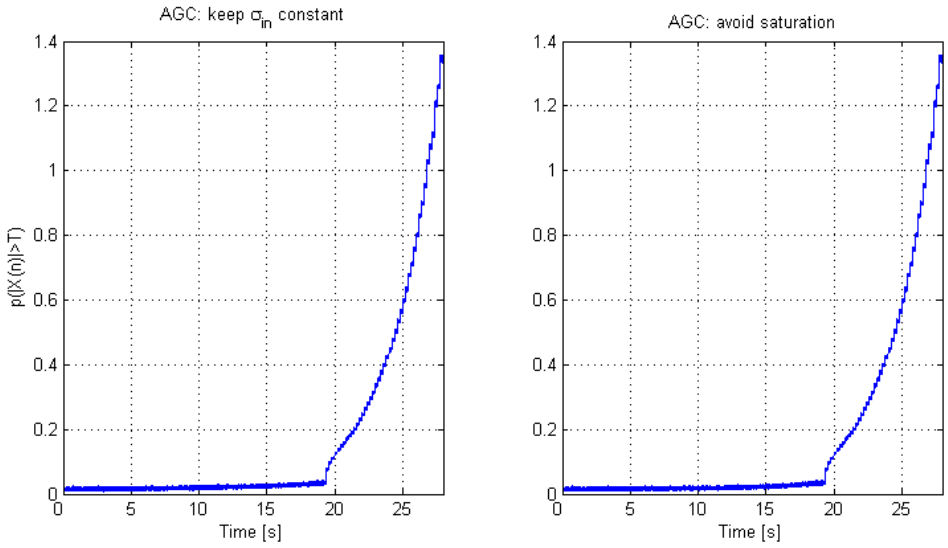

Figure 7.44.: Percentage of detected discrete frequencies exceeding the threshold $T_{freq}(p_{fa} = 1\%)$ for a 14 *bit* AGC/ADC, using a Hanning data window function to reduce spectral leakage.

are shown here.

Obviously, the receiver lost lock after ≈ 23 s of processing which is equivalent to an ISR of 113 dB, but for this point in time / for this ISR, the frequency excision techniques zeroed about 35 % of the signal's bandwidth, see Figure 7.44. Hence the real loss of lock was estimated occur at an ISR of about ≈ 107 dB, for this ISR about 20 % of the signal's bandwidth was zeroed, but the receiver was still able to proceed tracking. The decrease in the signal quality for times > 19.4 s, i.e. for ISRs > 95 dB, originates form the increased spectral excision and saturation effects in the ADC.

7.9.2. Final Performance Test of Frequency Excision Techniques with a 14-Bit AGC/ADC

Last but not least the performance of the frequency excision techniques in combination with a 14 *bit* AGC/ADC was investigated on a CW-RFI source, that varies its power strongly within a short duration of time. In this context the DLR GNSS Software Signal Generator was used to produce a GPS signal which was superposed with noise and two RFI sources, with the arbitrarily chosen carrier frequencies -50 Khz and 850 Hz apart f_{if}. The ISR variations of these signals versus time are shown in Figure 7.46.

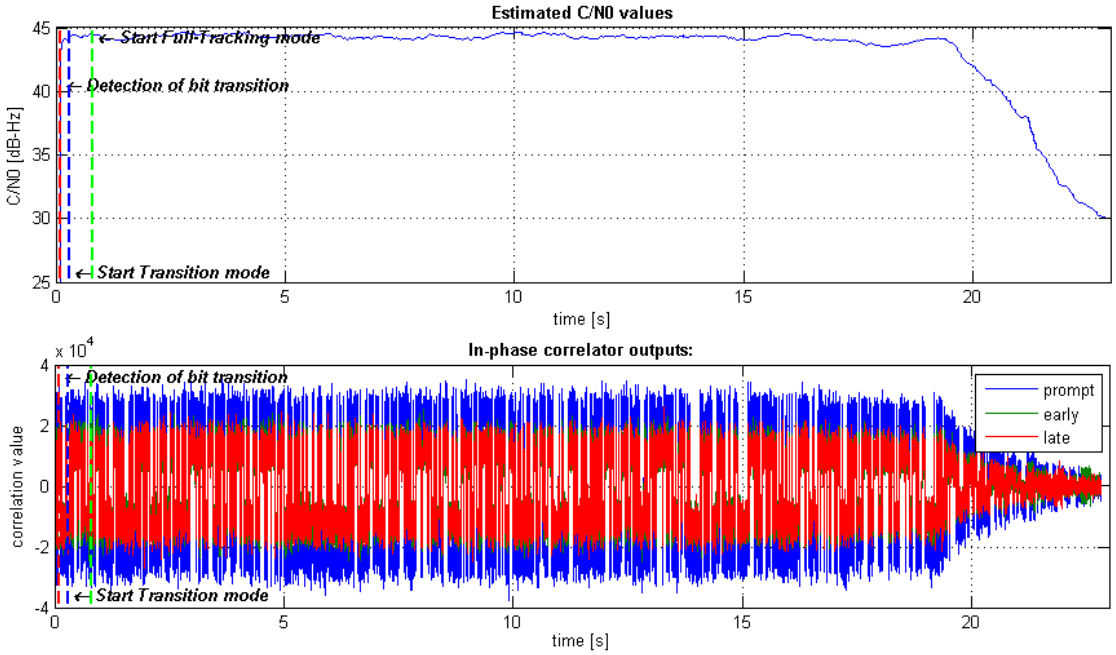

Figure 7.45.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 14 *bit* AGC/ADC in combination with frequency excision techniques, using overlapped Hanning data windows, under the impact of an increasing CW-RFI power

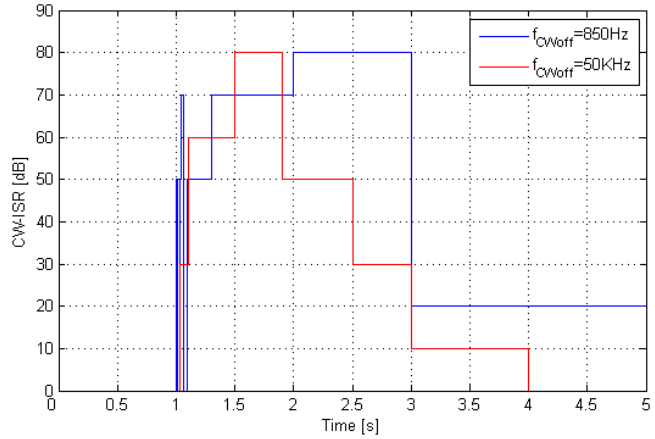

Figure 7.46.: Performance test: ISR variations of CW-RFI versus time.

Due to these drastic CW power fluctuations up to the order of 70 dB within 10 ms the AGC has to adapt its gain fast. Figure 7.47 shows the change in gain on the left and the percentage of discrete frequencies exceeding the spectral detection threshold $T_{freq}(p_{fa} = 1\ \%)$ versus time on the right. The objective of the AGC implemented was to avoid saturation in the ADC and to be fast, therefore it made use of a look-up-table. For the frequency excision techniques overlapped Hanning windows were used to reduce spectral leakage in the FFT computation.

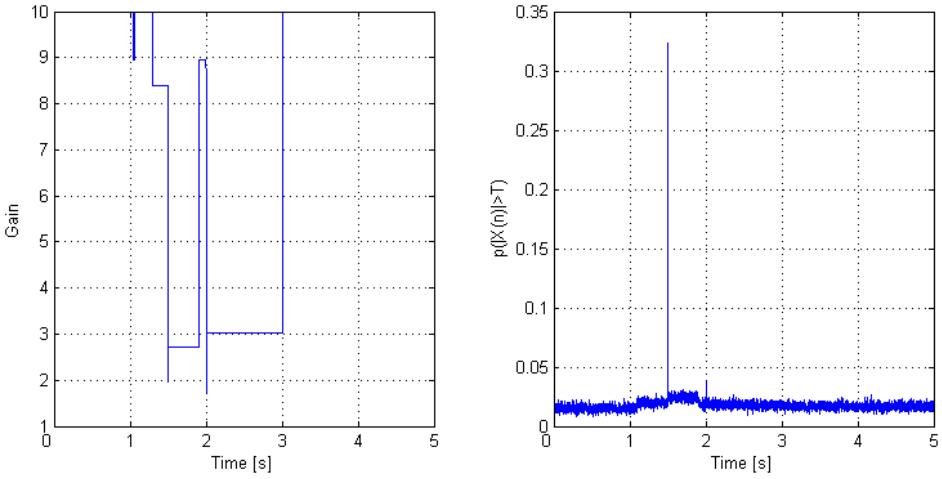

Figure 7.47.: Performance test: Change in gain and percentage of detected discrete frequencies exceeding the spectral threshold of $T_{freq}(p_{fa} = 1\%)$.

As one can see, the AGC can reduce its gain within a time interval of 1 ms completely. Peaks below the optimum gain arise due to saturation effects in the ADC, which lead to issues in the computation of the ADC output variance. Hence it needs another 1 ms to determine the optimum gain in the look-up-table. But anyway, already in the first ms after the gain was reduced, the frequency excision can deliver valid interference free data because to that time the ADC was no more in saturation. Hence those peaks in the percentage of detected discrete frequencies violating the spectral threshold are only 1 ms long and are the result of ADC saturation effects. Apart those peaks the percentage of spectral threshold violations stays in the order of the pre-defined false detection rate. The increase of the detection rate between the first and third second of processing is the consequence of additional quantization noise that is introduced to the quantized samples when the gain is decreased.

Next the quality of the signal freed of interference was investigated in the DLR Software Receiver, the results are shown in Figure 7.48.

As one can see, the frequency excision techniques and the AGC were able to reject

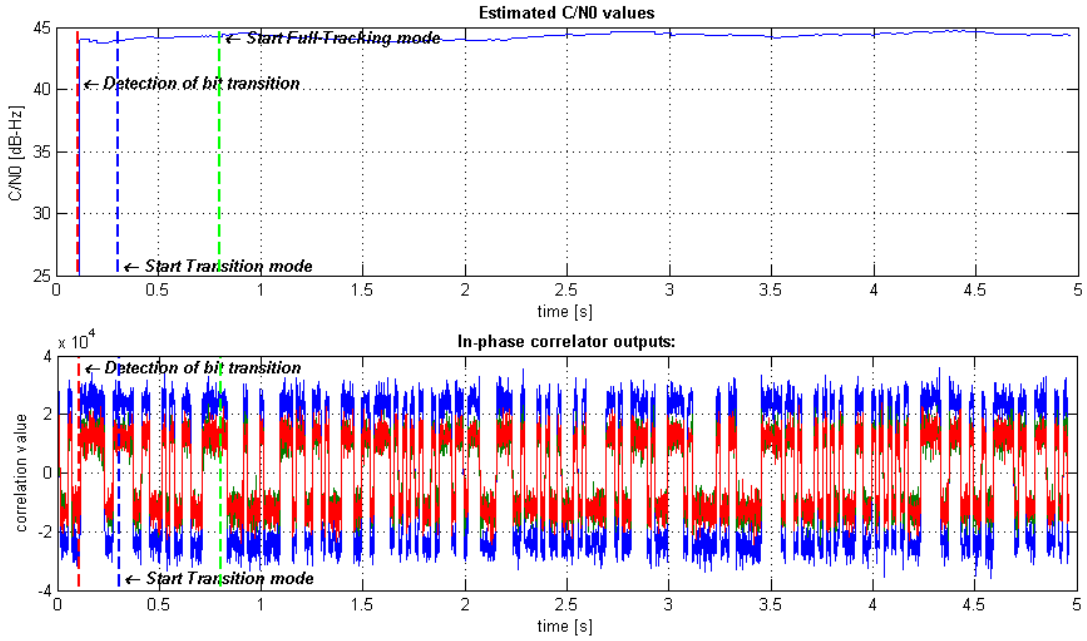

Figure 7.48.: Above: C/N_0 at the receiver output; Below: Correlator output; all for a 14 *bit* AGC/ADC in combination with frequency excision techniques, using overlapped Hanning data windows, under the impact of strongly varying CW powers

the interference properly, so the receiver does not suffer under the impact of these strong CW power fluctuations, hence the achieved C/N_0 and the signal quality at the correlator output maintained approximately constant.

Notice: Although the achieved signal quality at the correlator output maintained approximately constant for all gain settings, the signal quality at the circuit output is of course worsen due to the reduced gain and in connection with that the decreased signal resolution. So when on the obtained output data also digital beam forming shall be performed, some issues may arise because of these changes in signal quality.

7.10. Simulations on Temporal Blanking Techniques with a 14-Bit AGC/ADC

This section examines techniques for the detection and mitigation of pulsed RFI in the time domain. In this context especially the two different pulsed RFI detection strategies presented in Section 5.3 are compared against each other. For the simulations on temporal blanking techniques presented here the circuit illustrated in Figure 6.2 was used.

7.10.1. Performance Evaluation of Blanking Techniques dealing with Pulsed-RFI

In order to evaluate the performance of the developed temporal blanking techniques under worst case conditions the file depicted in Section 7.2.2, containing a DME interference with a PSR of 60 dB and a $dct = 10$ %, was driven to the AGC/ADC input of the circuit presented in Figure 6.2.

The total available gain was set according to the ratio $k_{opt} = 3$, to guarantee a 6 bit initial loading of the 14 bit ADC loading.

In allusion to the initial ADC loading, the digital detection threshold for a sample-by-sample interference detection was set according to the $6st$ bit, i.e. the digital detection threshold is $T = 2^6 - 1 = 63$. All sample exceeding this threshold were blanked, Figure 7.49 shows this detection approach in the time domain and on the ADC loading for the RFI free case.

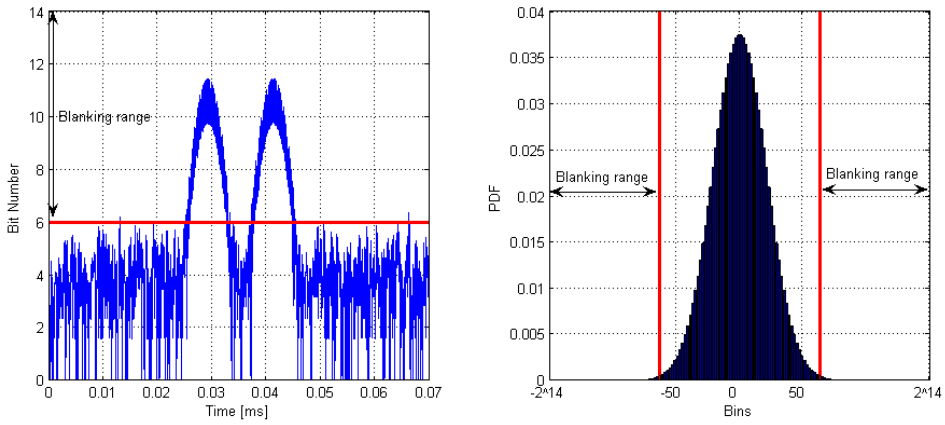

Figure 7.49.: Blanking threshold in a 14 *bit* ADC, for pulsed RFI detection on sample basis

As one can see, using this detection / blanking approach without any memory leads to the fact that also some valid samples are zeroed and those samples right before and after a DME pulse are not declared to be corrupted by the interference as they not exceed that threshold.

Hence, in Section 5.3 a different pulse detection technique was proposed, that performed the identification of pulsed RFI on the basis of power fluctuations in a shift register. As already explained, this strategy has the benefit that it can identify the start and end of a pulse so that also those samples right before and after threshold violation can be zeroed but not those that exceed the threshold due to fluctuations in the receivers noise floor.

For the detection of pulsed RFI on the basis of temporal power fluctuations the circuit presented in Figure 5.4 was used, thereby it had a total shift register length of $N_{reg} = 20$.

The detection threshold was set in allusion to the initial ADC loading, i.e. 20 adjacent samples had to be quantized at least in the ± 63 bin, so the digital threshold was computed according to Equation 5.6 as follows

$$T_{Pulse} = N_{Reg} \cdot (2^{bit_{res}} - 1)^2 = 20 \cdot (2^6 - 1)^2 = 79380.$$

For this detection threshold a excellent performance for the detection of pulsed RFI was achieved. Figure 7.50 shows the computed energy in the shift register when a pulse occurs and the detection threshold T_{Pulse}.

As one can see, using this approach the false detection rate $p_{fa} \rightarrow 0$ and one can clearly determine the start and end of such pulses. Another parameter jet to discuss

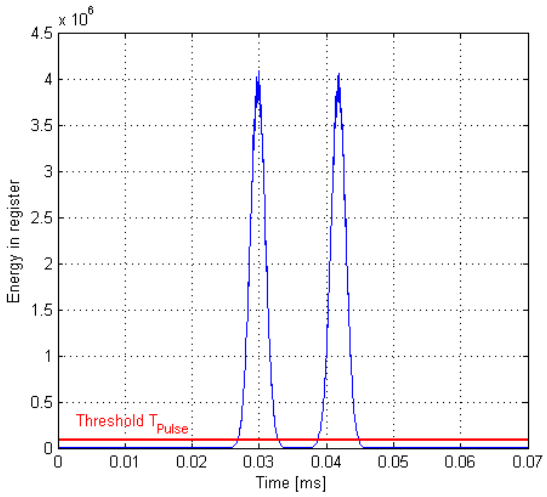

Figure 7.50.: Detection of pulsed RFI by monitoring power fluctuations in a shift register.

is the number of samples that shall be blanked right before and after a threshold violation. Simulations showed that the performance is best when about 15 samples right before and after a threshold violation are zeroed.

Next the performance of the different detection strategies are compared against each other. Therefore Figure 7.51 shows the circuit's output data for both strategies. Thereby, the first plot correspond to the quantized DME-pulse-pair over the complete 14 *bit* ADC range, the second one shows the obtained output data when blanking is performed on a sample-by-sample basis and the third one correspond to the case when a shift register was used to detect and mitigate pulsed interference.

As one can see, the approach that monitors temporal power fluctuations for the pulsed RFI detection is able to blank all corrupted samples. Next the output data of the blanking device was driven to the AGC and the DLR Software receiver. The gain curves are not depicted here, because they stayed rather constant due to the blanking process.

The achieved C/N_0 for both techniques is depicted in Figure 7.52, the plot above correspond to blanking on a sample-by-sample basis, the other one corresponds to blanking on the basis of temporal power fluctuations detection.

As one can see, the achieved C/N_0 is slightly better ($\approx 0.4\ dB$) for the techniques that monitors temporal power fluctuations and also zeros those sample directly before and after a threshold violation. Comparing the obtained results with those

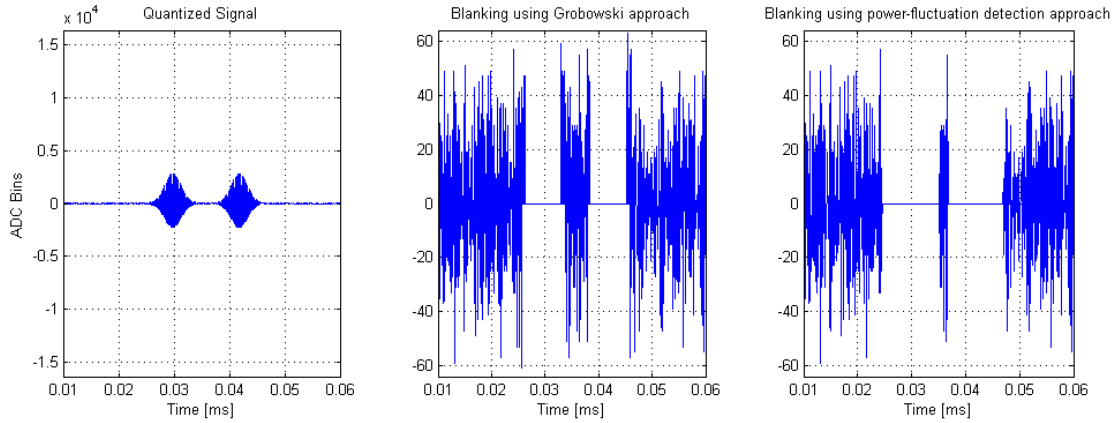

Figure 7.51.: Mitigation of pulsed RFI in the time domain using different methods; Left: Quantized DME-pulse-pair for a 14 Bit ADC; Middle: Blanking on sample-by-sample basis; Right: Blanking on the basis of energy fluctuations

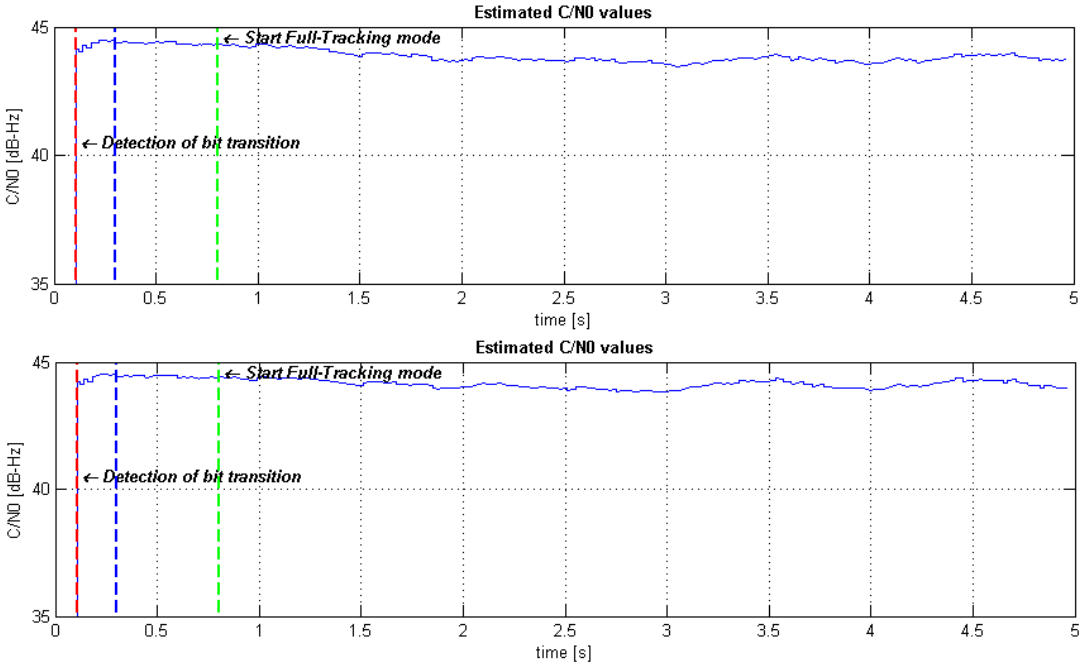

Figure 7.52.: Received C/N_0 for different blanking techniques; Above: Blanking on sample-by-sample basis; Below: Blanking on the basis of temporal power fluctuations detection

for the GPS C/A code in Section 7.4.2 and the frequency excision techniques in Section 7.8.2, it catches one's eye that temporal blanking is the best approach to mitigate the impact of pulsed RFI on the correlator.

This section investigated the performance of blanking techniques in a 14 *bit* ADC with a 6 *bit* initial loading. It was shown that temporal blanking is the best technique to reject pulsed RFI. Furthermore it was shown that blanking in combination with a shift register that monitors power fluctuations in the time domain has a better performance than interference detection and mitigation on a sample-by-sample basis. But however, the results are nearly same, so for the case that one has only to deal with pulsed RFI and noise, one should rather implement the sample-by-sample approach, because it does not need memory to detect pulses, nor it is necessary to determine their start and end. But when one also has to deal with WB and/or CW-RFI one should probably use the other method, because it can better distinguish between the different RFI types in the time domain.

8. Conclusions

8.1. Summary

The objective of this diploma thesis was the development of pre-correlation interference mitigation techniques for a GNSS receiver. Since these developed algorithms shall be implemented in the real world DLR Galileo receiver, some pre-defined parameters were given that respects specifics of the hardware on which these algorithms shall work. So, a ideal 20 dB AGC and a ideal 14 bit ADC were available for the adaptive A/D-conversion, gain steering should be performed on the ADC output data in 1 ms intervals. Further, the digitized signal should own a 6 bit resolution at the circuit output, provided that no RFI is present. Other specifics such as the circuit's complexity did not exist, hence it was possible to take use of the full advantages of the software defined radio.

Based on these given specifics, AGC steering strategies and interference detection and mitigation techniques were developed.

The AGC steering was performed with look-up-table because with this it is able to reduce the gain completely with 1 ms of ADC output data.

Interference detection was performed in the frequency and time domain as well as on the PDF of the ADC bins distribution. In this context also the Chi-Square Test was investigated, but due to its lack that it cannot determine the specifics of the interference, which there are frequency or point in time when it occurs, more complicated techniques had to be developed.

Therefore two different interference detection and mitigation strategies were developed taking place in the frequency as well as in the time domain.

Of major concern was thereby the development of frequency excision to mitigate the impact of narrowband and pulsed RFI in the frequency domain. Thereby it was shown that the best performance of these techniques is achieved when overlapped Hanning data windows are used to reduce spectral leakage in the discrete spectrum

computation. A simulative performance evaluation showed that these techniques are able to mitigate the impact of CW-like interference up to an ISR of $\approx 83dB$, when only a 14 *bit* ADC is implemented, and up to an ISR of ≈ 107 dB, when also an AGC is available. For both of these ISRs the ADC was oversteered, but simulations showed that the frequency excision techniques and the GPS C/A-code can handle saturation effects in the ADC up to a certain level. Furthermore is was shown that these techniques are also able to mitigate the impact of pulsed RFI on the correlator, but the achieved C/N_0 suffered a degradation of ≈ 3 dB.

Hence different techniques were developed to mitigate pulsed RFI in the time domain. In this context temporal blanking was introduced, which was performed on a sample-by-sample basis as well as monitoring power fluctuations in the received signal. Both of these techniques showed an excellent performance, so that the achieved C/N_0 was only slightly degraded compared to the RFI free case. But for a implementation in the DLR Galileo receiver one should use that method that monitors power fluctuations in the received signal, because it can identify the start and end of a pulse, and so perform a better temporal blanking (the achieved C/N_0 is about 0.4 dB as for the sample-by-sample technique) and further one can better distinguish between pulsed interference and other interference types in the time domain.

8.2. Outlook

Simulations showed that the frequency excision techniques are the best approach to mitigate narrowband RFI and temporal blanking techniques are the best approach to mitigate pulsed RFI. Hence the overall performance of the receiver can be increased when these techniques are combined in a hybrid. Thereby one should make use of that temporal blanking technique that monitors power fluctuations in the received signal, because it can better distinguish between pulsed interference and other interference types in the time domain.

Further it was indicated that the SNR of the quantized signal for a 6 *bit* initial ADC loading is about 17 dB in the magnitude spectrum, see Figure 7.27. But when the gain is set to 0 dB the SNR is degraded to 7 dB due to quantization noise. But however, the threshold adaption and spectrum computation still worked fine, hence an AGC with a range of 30 dB might be implemented to decrease saturation in the ADC even beyond an ISR of 95 dB, which was the maximum ISR for which no saturation in the ADC occurred when using a 20 dB AGC, and so increase the over all receiver performance.

As mentioned earlier, the AGC steering approach is well known not to be optimal,

because it assumed a Gaussian distribution of the ADC bins. Hence the AGC reduced it gain for ISRs $> 60dB$, for which the ADC was not even close to a full loading. Hence, regarding that CW RFI is the most likely interference type that is able to produce such high and constant RFI powers, an adaption of the expected ADC bins distribution to that a sine should bring to a better gain steering strategy.

The FFT here was always computed of vectors containing quantized samples of a time interval of 1 ms, hence each of these FFTs had $2 \cdot 10^4$ points. But for the implementation the designer should regard that the FFT has a butterfly like structure of its algorithm, with which the number of computations is reduced, because of that the number of samples used for the FFT computation should be of $N = 2^k$ to minimize computation time. In addition, more investigations about the temporal vector length should be done, because less time, i.e. less samples reduces the complexity of the FFT computation, but also decreases the spectral resolution. Hence the designer should search for a trade of between complexity of the FFT computation and the necessary spectral resolution for frequency excision techniques to work properly.

A. Developed Program Codes

A.1. Look-up-Table

```
% Generate look-up table for AGC

% Parameters


L                = 1;     % ^= highest quantization threshold,
                          % e.g. ADC range is between -1V and +1V

nBitsADC         = [6 8]; % [a b]
                          % a = number of initially used ADC bits
                          % b = number of bits for dynamic range
AGC_gain_init    = 20;    % total AGC gain
num_samples      = 10000;   % number of samples to compute statistics
                          % for each value in the table
FlagAGCfullRange = 0;     % if 1, adjust gain to avoid saturation
                          % if 0, adjust gain to keep input power const.


%--------------------------------------------------------------------
%--------------------------------------------------------------------
% Compute the ADC parameters

% Highes ADC value
maxADCBin = 2^(nBitsADC(1) + nBitsADC(2))-1;

% number of total ADC bins
nLevelsADC = 2^(nBitsADC(1) + nBitsADC(2));

% ADC quantization thresholds
ADC_thresholds = linspace(-L, L, nLevelsADC-1);
```

```
% ADC output range (ADC bins)
ADC_range = linspace(-maxADCBin, maxADCBin, nLevelsADC);

%-------------------------------------------------------------------------
% find the maximum threshold for the resolution bins
% see Basttide p. 171 ff.;
% 'abs(ADC_thresholds(1)-ADC_thresholds(2))' is the quatization step size
if FlagAGCfullRange
    L_new = L;
else
    L_new = (2^(nBitsADC(1) - 1) - 1) * ...
        abs(ADC_thresholds(1) -  ADC_thresholds(2));
end;

%--- determine ADC resolution range factor [Bastide Ph.D thesis, p. 63]
switch nBitsADC(1)
   case {1, 2}
       kADC = 0.986;
   case 3
       kADC = 1.731;
   case 4
       kADC = 2.291;
   case 5
       kADC = 2.7225;
   case 8
       kADC = 3.0;
   otherwise
       kADC = 3.0;
end


%-------------------------------------------------------------------------
%-------------------------------------------------------------------------



isr     = [-inf -20:80];

isr_l   = 10.^(isr/20);

% generate noise
```

```
n       = randn(length(isr),num_samples);

s       = [];

% generate signal, noise + gaussian interference with different powers
for i=1:length(isr)
    rfi     = randn(1,num_samples);
    rfi     = rfi / std(rfi) * isr_l(i);
    s(i,:)  = n(i,:) / std(n(i,:)) + rfi;
end

% generate noise
n       = randn(length(isr),num_samples);

% compute front end gain
gain_front_end = L_new / kADC / std(n(1,:)) / 10^(AGC_gain_init/20);

% linear initital agc gain
gain_init = 10^(AGC_gain_init/20);

% vector of different gain values
gain = [zeros(1,length(isr))];

% Compute gain curve
for i=1:length(isr)
    gain(i) = gain_init / std(s(i,:));
end;


% Reduce the gain range, i.e when the gain has decreased to 0dB delete
% all other points and also make look up table only up to thispoint
gain = [gain(find(gain > 1)) 1];  % this are the possible gain settings


% computet output variance
sigma_out = zeros(length(gain),length(gain));


h = waitbar (0,'Computing gain look-up table');

for i = 1:length(gain)
```

```
    waitbar(i/length(gain));

    for k = 1:length(gain)

        signal = gain(i) * gain_front_end * s(k,:);

        [index, output]= quantiz(signal, ADC_thresholds, ADC_range);

        sigma_out(i,k) = std(output);
    end;
end;

close(h);

gain_table = gain;
sigma_table = sigma_out;
% Save look up table
save('gain_look_up_table','gain_table','sigma_table');
```

A.2. How to Execute a Program

The developed program codes are developed in that manner that they can be directly be implemented in the DLR GNSS Signal generator.

The necessary program code settings can be made in the "generatorSettings" file.

When one wants to run the program code separately do the following:

```
genResults.sDataFile{1} = 'Inputfile';

genSettings  = generatorSettings;

genResults   = programname(genSettings, genResults);
```

A.3. The Chis-Square-Test

A.3.1. Settings

```
function genSettings = generatorSettings
% Setting the GNSS generator parameters
%
% Usage:
%   genSettings = generatorSettings
%
% Inputs: no inputs needed
%
% Outputs:
%   genSettings = structure containing generator setting parameters
%

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%Thorsten

genSettings.Nant = 1;
genSettings.fs  = 2e7;           % sampling frequency [Hz]

genSettings.AGC_gain_init = 20;  % [dB] AGC gain at it's start
genSettings.look_up_table_path = 'Data\gain_look_up_table60';

genSettings.nBitsADC = [6 0];    % [a b]
                                 % a = number of initially used ADC bits
                                 % b = number of bits for dynamic range
genSettings.nBits_SoftQuant = 6; % number of bits for software quantization
genSettings.L = 1;               % Maximum ADC threshold


genSettings.get_statisticcs_time = 0.1; % time in which statistics
                                        % are computed
genSettings.sampleInterval = 2e4; % number of samples per block for
                                  %interference detection and AGC setting
genSettings.outDataFormatADC = 'int8';% Data Format after the ADC


%--- Output data format
```

```
genSettings.outDataType   = 'real';   % possible options: {'real', 'I&Q'}
genSettings.outDataFormat = 'double'; % possible options: {'int8',
                                      % 'schar', 'double'}
end
```

A.3.2. Program Code

```
function genResults = chi_square_test(genSettings, genResults)
% Chi-Square Test
% Usage:
%       chi_square_test(genSettings, genResults)
%
% Inputs:
%   genSettings = structure containing generator setting parameters
%   genResults  = structure containing parameters of the generated signal
%
% Usage:
%   [flagOK, genResults] = produceSignal(genSettings)
%
% Inputs:
%   genSettings = structure containing generator setting parameters
%
% Outputs:
%       a file containing the 'digitized data'


% Thorsten Lotz, Diploma Thesis at DLR

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%Parameters
Nant = genSettings.Nant;                      % number of antenna elements
sampleInterval = genSettings.sampleInterval; % number of elements used
                                              % for interference
                                              % detection and AGC setting

fs = genSettings.fs;
```

```
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%--- Open the files obtained from the GNSS signal generator
for m = 1 : Nant


    %Open the files obtained from the GNSS signal generator
    [inputfile(m), message] = fopen(genResults.sDataFile{m}, 'r');


    if (inputfile(m) < 0)
        error([' Error opening the data file ', genResults.sDataFile{m}])

    else
        fseek(inputfile(m), 0, 'bof');     %Set pointer to first element

        %data obtains the incomming signal
        [signal, count] = fread(inputfile(m), sampleInterval,...
            genSettings.outDataFormat);

        if (count < sampleInterval)
        % The file is too short
        error('Could not read enough data from the data file');
        end;

    end;
end;


%-- determine some loop parameters
switch genSettings.outDataFormat
    case 'int8'
        data_type_size = 1;
    case 'schar'
        data_type_size = 1;
    case 'double'
        data_type_size = 8;
    otherwise
        data_type_size = 1;
end
```

```
% determine the total number of Samples in the File
fseek(inputfile(1), 0, 'eof');
total_number_of_samples = ftell(inputfile(1)) / data_type_size;
fseek(inputfile(1), 0, 'bof');



% number of blocks for the for loop
nBlocks = floor(total_number_of_samples / sampleInterval);
% rest of the Samples in the file, not quantized in the loop
sampleRest = floor( mod(total_number_of_samples, sampleInterval) );


%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Compute the ADC parameters

% =^ highest quantization threshold, e.g. ADC range is between -2V and +2V
L = genSettings.L;

maxADCBin = ceil(2^(genSettings.nBitsADC(1) + genSettings.nBitsADC(2)))-1;

% number of total ADC bins
nLevelsADC = ceil(2^(genSettings.nBitsADC(1) + genSettings.nBitsADC(2)));

% ADC quantization thresholds
ADC_thresholds = linspace(-L, L, nLevelsADC-1);

% ADC output range (ADC bins)
ADC_range = linspace(-maxADCBin, maxADCBin, nLevelsADC);


%--- determine ADC resolution range factor [Bastide Ph.D thesis, p. 63]
switch genSettings.nBitsADC(1)
   case {1, 2}
       kADC = 0.986;
   case 3
       kADC = 1.731;
   case 4
       kADC = 2.291;
   case 5
```

```
        kADC = 2.7225;
    case 8
        kADC = 3.0;
    otherwise
        kADC = 3.0;
end

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% find the maximum threshold for the resolution bins
% see Basttide p. 171 ff.;
% 'abs(ADC_thresholds(1)-ADC_thresholds(2))' is the quatization step size
L_new       = (2^(genSettings.nBitsADC(1) - 1) - 1)...
               * abs(ADC_thresholds(1)-ADC_thresholds(2));

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Get some Statistics of the Interference free Signal
% It is assumed that within the first genSettings.get_statisticcs_time sec
% no Interference is present
%-------------------------------------------------------------------------
iterations_for_statistics = floor(genSettings.get_statisticcs_time...
                            * fs / sampleInterval);

h               = waitbar (0, 'Computing signal statistics');
h_counter       = 0;

%- Standart deviation of the signal
std_signal      = zeros(1, iterations_for_statistics);

%- Target variance of quantized signal
var_target      = zeros(1, iterations_for_statistics);

%- Register for detection of temporal power fluctuations
histin = zeros(1,length(ADC_range));

for iter = 1:iterations_for_statistics

    h_counter           = h_counter + 1;              % Waitbar
    waitbar(h_counter / iterations_for_statistics);

    [signal, count]     = fread(inputfile(m), sampleInterval,...
```

```
                          genSettings.outDataFormat);

    std_signal(iter)    = std(signal);

    % Power adjustenment of the incomming signal according to the ADC range
    % only within the first 100 ms std(signal) is assumed to be known!
    signal_new          = signal / std(signal) * L_new / kADC;

    % Quantization of the test statistic
    [index, quant]      = quantiz(signal_new, ADC_thresholds, ADC_range);

    % Target variance for AGC setting and Blanking
    var_target(iter)    = std(quant);
    histin = histin + hist(quant,ADC_range);

end;
genResults.histin = histin / sum(histin);

close (h);
fseek(inputfile(m),0,'bof');

%-----------------------------------------------------------------------
%- Set Front-End Gain
% It is assumed that in the first 100 ms of the Signal no interference is
% present and one knows the standart deviation of the incomming signal.
% So the total gain of the receivers front end is set according to kADC
% and genSettings.AGC_gain_init
gain_front_end  = L_new / kADC / 10^(genSettings.AGC_gain_init/20)...
                  / (sum(std_signal) / iterations_for_statistics);

%- Target variance for AGC setting and Blanking
var_target      = sum(var_target) / iterations_for_statistics;

%-----------------------------------------------------------------------

%-----------------------------------------------------------------------
%-- END OF ADC SETTINGS
%-----------------------------------------------------------------------


%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
```

```
%- Parameter computation for Histogram based RFI detection and AGC setting
%-------------------------------------------------------------------------

%- Compute theoretical Bin Frequencies
BinFreq_nom     = zeros(1,(2^(sum(genSettings.nBitsADC) - 1)));
%- Thresholds to compute bin Frequencies
L_nom           = linspace(0,1,(2^(sum(genSettings.nBitsADC) - 1)));
L_nom           = [L_nom Inf];
for i = 1:2^(sum(genSettings.nBitsADC) - 1)

    BinFreq_nom(i)  = 0.5 * erfc(L_nom(i) / sqrt(2) * kADC)...
                    - 0.5 * erfc(L_nom(i + 1) / sqrt(2) * kADC);

end;

%- the bin loading is symmetric due to zero mean
BinFreq_nom     = [fliplr(BinFreq_nom) BinFreq_nom];

%-------------------------------------------------------------------------
%- End of Parameter computation for Histogram based RFI detection and AGC
%  setting
%-------------------------------------------------------------------------


%-- load look-up-table for AGC
load(genSettings.look_up_table_path, 'gain_table', 'sigma_table')



%-------------------------------------------------------------------------
%- Quantization and Interference mitigation
%-------------------------------------------------------------------------

h           = waitbar (0, 'Computing quantized samples');
h_counter   = 0;
epsilon     = 0; % difference between target and present variance
gain_init   = 10^(genSettings.AGC_gain_init/20);
gain        = gain_init;
gain_pos    = 1;

% for test purpose:
ggain       = zeros(Nant,nBlocks );
```

```
for m = 1 : Nant

   for k = 1 : nBlocks

       % test: save current gain
       ggain(m,k)       = gain;

       h_counter        = h_counter + 1;              % Waitbar
       waitbar(h_counter / (nBlocks * Nant));

       signal           = gain_front_end * fread(inputfile(m),...
                          sampleInterval, genSettings.outDataFormat);

       %- Signal level adjustenment
       signal_new       = signal * gain;

       %- Quantization
       [indx, quant]    = quantiz(signal_new, ADC_thresholds, ADC_range);

       BinFreq_cur      = hist(quant, ADC_range) / sampleInterval;

       % Chi-Square Test
       BinT(k)          = sum((BinFreq_cur - BinFreq_nom).^2 ./...
           (BinFreq_cur + BinFreq_nom));


       % AGC using a look-up-table
       % table-search algorithm
       % estimate correct gain on basis of output variance
       quant_to_AGC = quant;
       a = find(std(quant_to_AGC) > sigma_table(gain_pos,:),1,'last');
       b = find(std(quant_to_AGC) < sigma_table(gain_pos,:),1,'first');

       if isempty(a)
           gain_pos = b;
       elseif isempty(b)
           gain_pos = a;
       else
           if (abs(std(quant_to_AGC) - sigma_table(a,:))) >...
                   (abs(std(quant_to_AGC) - sigma_table(b,:)))
               gain_pos = b;
```

```
            else
                gain_pos = a;
            end
        end

        gain = gain_table(gain_pos);

    end;

end;

fclose all;
close(h);
clear h_counter;

genResults.gain         = ggain;
genResults.BinT         = BinT;

end
```

A.4. Frequency Excision Techniques

The program code shown here describes the circuits of Figure 6.3 and 6.6.

A.4.1. Settings

```
function genSettings = generatorSettings
% Setting the GNSS generator parameters
%
% Usage:
%   genSettings = generatorSettings
%
% Inputs: no inputs needed
%
% Outputs:
%   genSettings = structure containing generator setting parameters
%
```

```
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%Thorsten

genSettings.Nant            = 1;
genSettings.fIF             = 4e6;   % intermediate frequency [Hz]
genSettings.fs              = 2e7;   % sampling frequency [Hz]
genSettings.filtCutoffFreq  = 2.0e6; % filter cut-off frequency [Hz],
                                     %i.e. aproximate single-side bandwidth

genSettings.AGC_gain_init   = 20;    % [dB] AGC gain at it's start,
                                     % and also the total range

genSettings.FlagAGCon       = 1;     % here one can set whether or not an
                                     % AGC shall be used, if =0, the gain
                                     % will have the constant value
                                     % AGC_gain_init


genSettings.Flag_use_look_up_table = 1; % choose wheter to use a look-up-
                                        % table (1) or a PID controller (0)

% for look up table only
genSettings.look_up_table_path  = 'Data\look_up_table86_full.mat';

% for PID controller
genSettings.FlagAGCfullRange    = 1; % if 1 then the target gain is
                                     % the maximum signal standard
                                     % deviation for the complete ADC range
genSettings.AGC_Ki              = -0.001;  % for PI controller


% Pulse detection
genSettings.FlagDMEdetectionOn = 0; % (0) no DME detection for AGC steering

genSettings.dme_threshold           = 20 * (2^6-1)^2;

genSettings.length_shift_register   = 20;   % Shift Register length

genSettings.increase_blanking_range  = 15;  % Number of samples which will
                        % also be blanked before and  after DME threshold
                        % violation, respectively necessary for AGC
```

```
genSettings.nBitsADC              = [8 6]; % [a b]
                          % a = number of initially used ADC bits
                          % b = number of bits for dynamic range

genSettings.nBits_SoftQuant = 6;   % number of bits for software quantization


genSettings.FFT_window_type = 'shan';   % Window type for FFT computation
                % possible: hann, rect or shan for sliding hanning window

genSettings.L = 1;                      % Maximum quantization threshold
                                        % of A/D converter

genSettings.get_statisticcs_time = 0.1; % time in which statistics shall
                                        % be computed


genSettings.sampleInterval = 2e4;       % number of samples per block for
                                        % interference detection/mitigation
                                        % and AGC setting

genSettings.outDataFormatADC = 'int8';  % Data Format after the ADC


%--- Output data format
genSettings.outDataType   = 'real';   % possible options: {'real', 'I&Q'}
genSettings.outDataFormat = 'double'; % input file type


%--- Frequency excision techniques
genSettings.FFTpoints = genSettings.sampleInterval; % Number of points
                                                    %used for FFT computation

genSettings.false_alarm_detection_propability = 0.01;
   % [0..1] Neumann Pierson Approach: for computation of interference
   % detection threshold depending on the signal when no RFI is present
end
```

A.4.2. Program Code

```
function genResults = freq_ex(genSettings, genResults)
% Quantization and pre-correlation Interference Mitigation of the generated
% GNSS signal
%
% Usage:
%       freq_ex_sliding_hann(genSettings, genResults)
%
% Inputs:
%   genSettings = structure containing generator setting parameters
%   genResults  = structure containing parameters of the generated signal
%
% Usage:
%   [flagOK, genResults] = produceSignal(genSettings)
%
% Inputs:
%   genSettings = structure containing generator setting parameters
%
% Outputs:
%       a file containing the 'digitized data'


% Thorsten Lotz, Diploma Thesis at DLR

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%Parameters
Nant = genSettings.Nant;                      % number of antenna elements
sampleInterval = genSettings.sampleInterval; % number of elements used
                                              % for interference detection
                                              % and AGC setting

fs = genSettings.fs;


%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%--- Open the files obtained from the GNSS signal generator
for m = 1 : Nant
```

```
    %Open the files obtained from the GNSS signal generator
    [inputfile(m), message] = fopen(genResults.sDataFile{m}, 'r');


    if (inputfile(m) < 0)
        error([' Error opening the data file ', genResults.sDataFile{m}])

    else
        fseek(inputfile(m), 0, 'bof');       %Set pointer to first element

        %data obtains the incomming signal
        [signal, count] = fread(inputfile(m), sampleInterval,...
            genSettings.outDataFormat);

        if (count < sampleInterval)
        % The file is too short
        error('Could not read enough data from the data file');
        end;

    end;
end;



%--- Assign names for output data file
for n = 1 : Nant
    sDataFileADC{n} = ['adc_', num2str(sum(genSettings.nBitsADC)),'bit',...
        '_fr_ex_', genSettings.FFT_window_type, ...
        '_agc_', num2str(genSettings.FlagAGCon), ...
        '_dme_detec_', num2str(genSettings.FlagDMEdetectionOn), '_', ...
        genResults.sDataFile{n}((find(genResults.sDataFile{n} == '\')+1)...
        :length(genResults.sDataFile{n}))];
    outputfile(n) = fopen(['Results\', sDataFileADC{n}], 'w');
end



%-- determine some loop parameters
switch genSettings.outDataFormat
    case 'int8'
        data_type_size = 1;
```

```
    case 'schar'
        data_type_size = 1;
    case 'double'
        data_type_size = 8;
    otherwise
        data_type_size = 1;
end


% determine the total number of Samples in the File
fseek(inputfile(1), 0, 'eof');
total_number_of_samples = ftell(inputfile(1)) / data_type_size;
fseek(inputfile(1), 0, 'bof');


% number of blocks for the for loop
nBlocks = floor(total_number_of_samples / sampleInterval);
% rest of the Samples in the file, not quantized in the loop
sampleRest = floor( mod(total_number_of_samples, sampleInterval) );


%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Compute the ADC parameters

% =^ highest quantization threshold, e.g. ADC range is between -2V and +2V
L = genSettings.L;

maxADCBin = 2^(genSettings.nBitsADC(1) + genSettings.nBitsADC(2))-1;

% number of total ADC bins
nLevelsADC = 2^(genSettings.nBitsADC(1) + genSettings.nBitsADC(2));

% ADC quantization thresholds
ADC_thresholds = linspace(-L, L, nLevelsADC-1);

% ADC output range (ADC bins)
ADC_range = linspace(-maxADCBin, maxADCBin, nLevelsADC);


%--- determine ADC resolution range factor [Bastide Ph.D thesis, p. 63]
switch genSettings.nBitsADC(1)
```

```
   case {1, 2}
       kADC = 0.986;
   case 3
       kADC = 1.731;
   case 4
       kADC = 2.291;
   case 5
       kADC = 2.7225;
   case 8
       kADC = 3.0;
   otherwise
       kADC = 3.0;
end

%--- determine optimal ratio factor for software quantization
% [Bastide Ph.D thesis, p. 63]
switch genSettings.nBits_SoftQuant
   case {1, 2}
       kADC_soft = 0.986;
   case 3
       kADC_soft = 1.731;
   case 4
       kADC_soft = 2.291;
   case 5
       kADC_soft = 2.7225;
   case 8
       kADC_soft = 3.0;
   otherwise
       kADC_soft = 3.0;
end

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% find the maximum threshold for the resolution bins
% see Basttide p. 171 ff.;
% 'abs(ADC_thresholds(1)-ADC_thresholds(2))' is the quatization step size
L_new = (2^(genSettings.nBitsADC(1) - 1) - 1)...
        * abs(ADC_thresholds(1)-ADC_thresholds(2));

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% in order to reduce the total range of the ADC to only resolution range
```

```
% one needs to 'quantize' one more time, therefore compute some parameters
% of the second software quantization, the data output of the quantizer is
% given to the correlator
soft_quant_thresholds = linspace(-(2^genSettings.nBits_SoftQuant - 1),...
    2^genSettings.nBits_SoftQuant - 1, 2^genSettings.nBits_SoftQuant-1);
soft_quant_range = linspace(-(2^genSettings.nBits_SoftQuant - 1),...
    2^genSettings.nBits_SoftQuant - 1, 2^genSettings.nBits_SoftQuant);
%--------------------------------------------------------------------------
%-- END OF ADC SETTINGS
%--------------------------------------------------------------------------

%--------------------------------------------------------------------------
%- Different Window types for FFT computation
if genSettings.FFT_window_type == 'rect'
    window = ones(1,sampleInterval);
    flag_sliding_hann = 0;
elseif genSettings.FFT_window_type == 'hann'
    window = sin(0.5*2*pi*(1:sampleInterval)/sampleInterval) .^2;
    flag_sliding_hann = 0;
elseif genSettings.FFT_window_type == 'shan'
    window = sin(0.5*2*pi*(1:sampleInterval)/sampleInterval) .^2;
    flag_sliding_hann = 1;
else
    error('Window must be of type hann or rect');
end;

% Domain of the FFT where there is no Front-End filter
f_start = (genSettings.fIF - genSettings.filtCutoffFreq)...
    / genSettings.fs * genSettings.FFTpoints;
f_end   = (genSettings.fIF + genSettings.filtCutoffFreq)...
    / genSettings.fs * genSettings.FFTpoints;


%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Get some Statistics of the Interference free Signal
% It is assumed that within the first 100 ms no Interference is present
%--------------------------------------------------------------------------
iterations_for_statistics = floor(genSettings.get_statisticcs_time...
    * fs / sampleInterval);

h = waitbar (0, 'Computing signal statistics');
```

```
h_counter = 0;


%- Standart deviation of the undistored signal
std_signal_normal      = zeros(1, iterations_for_statistics);

%- Mean of the maginitude spectrum within the filter when agc is at its max
mean_magnitude_sig_spec_full_agc = zeros(1, iterations_for_statistics);

%- Mean of the maginitude spectrum within the filter when agc is at its min
mean_magnitude_sig_spec_min_agc = zeros(1, iterations_for_statistics);

%- Target variance of quantized signal
var_normal      = zeros(1, iterations_for_statistics);

%- Register for detection of temporal power fluctuations
shift_register  = zeros(1,genSettings.length_shift_register);

%- Register to save current position in look-up-table
gain_pos        = 1;

for iter = 1:iterations_for_statistics

    h_counter           = h_counter + 1;              % Waitbar
    waitbar(h_counter / iterations_for_statistics);

    [signal, count]     = fread(inputfile(m), sampleInterval,...
        genSettings.outDataFormat);

    std_signal_normal(iter)    = std(signal);

    %- Power adjustenment of the incomming signal according to the ADC range
    %  only within the first 100 ms, std(signal) is assumed to be known!
    signal_new          = signal / std(signal) * L_new / kADC;

    % Quantization of the test statistic
    [index, quant]      = quantiz(signal_new, ADC_thresholds, ADC_range);

    %-------------------------------------------------------------------
    %- Initializatuions for frequency excision techniques
    %- Windowing
    quant_window        = window.*quant;
```

```
sig_spec            = fft(quant_window, genSettings.FFTpoints);

%- Magnitude Spectrum
magnitude_sig_spec  = abs(sig_spec);

mean_magnitude_sig_spec_full_agc(iter) = mean(magnitude_sig_spec(f_start:f_end))


%-------------------------------------------------------------------
%-------------------------------------------------------------------
% compute signal spectrum when gain is in its lower border

signal_new          = signal / std(signal) * L_new / kADC * 10^(-genSettings.AGC

% Quantization of the test statistic
[index, quant]      = quantiz(signal_new, ADC_thresholds, ADC_range);

%-------------------------------------------------------------------
%- Initializatuions for frequency excision techniques
%- Windowing
quant_window        = window.*quant;

sig_spec            = fft(quant_window, genSettings.FFTpoints);

%- Magnitude Spectrum
magnitude_sig_spec  = abs(sig_spec);

mean_magnitude_sig_spec_min_agc(iter) = mean(magnitude_sig_spec(f_start:f_end));
%-------------------------------------------------------------------
%-------------------------------------------------------------------


%-------------------------------------------------------------------
%- Initializations for DME detection
%- Computition of the average signal power
if genSettings.FlagDMEdetectionOn
    tmp                 = [shift_register quant];
    power_in_register   = zeros(1,sampleInterval);

    for ii = 1:(sampleInterval)
        power_in_register(ii)    = std(tmp(ii:(ii +...
```

```
                genSettings.length_shift_register))).^2;
        end;
        mean_power_in_register(iter) = mean(power_in_register);

        %- init register for next run
        shift_register      = quant( (sampleInterval...
            - genSettings.length_shift_register + 1):sampleInterval );

        % Normal variance of incomming signal, necessary for
        % AGC setting and Blanking
        var_normal(iter)    = std(quant);
    end
    %- Initializations for DME detection
    %-------------------------------------------------------------------
end;
close(h);

%-------------------------------------------------------------------
%- Set Front-End Gain
% It is assumed that in the first 100 ms of the Signal no interference is
% present and one knows the standart deviation of the incomming signal.
% So the total gain of the receivers front end is set according to kADC and
% genSettings.AGC_gain_init
gain_front_end = L_new / kADC / 10^(genSettings.AGC_gain_init/20) /...
    (sum(std_signal_normal) / iterations_for_statistics);


%-------------------------------------------------------------------
%- CW detection threshold
% The Power Spectral Density follows a Rayleight distribution, because
% the real and imaginary part of the fft of a gaussian random process are also
% gaussian random processes, but the power spectral densiry is computed by
% |S(f)| = sqrt(RE{S(f)}^2 + IM{S(f)}^2) <= this process has a Rayleigt PSD.
% The detection threshold is computed by the cdf of the Rayleight distribution
spectra_detection_threshold      = sqrt(-2 *...
    log(genSettings.false_alarm_detection_propability));

% This value is only correct if var(|S(f)|) = 1. So we have adjust this
% threshold according to mean of the signal and sqrt(pi/2)
mean_magnitude_sig_spec          = sum(mean_magnitude_sig_spec_full_agc)...
    / iterations_for_statistics;
spectra_detection_threshold = spectra_detection_threshold...
```

```
    * sqrt(2/pi) * mean_magnitude_sig_spec;

%- End of computing parameters for frequency excision
%--------------------------------------------------------------------


% max amplification factor of software amplifier after real ADC


max_soft_agc = (sum(10*log10(mean_magnitude_sig_spec_full_agc)) -...
    sum(10*log10(mean_magnitude_sig_spec_min_agc))) / iterations_for_statistics;

max_soft_agc = 10^(max_soft_agc/20);



%--------------------------------------------------------------------
%- Parameters of DME detection
%- Normal variance of incomming signal
var_normal         = sum(var_normal) / iterations_for_statistics;

%- Variance of Signal before ADC
std_signal_normal  = sum(std_signal_normal) / iterations_for_statistics;


%--------------------------------------------------------------------
%- Blanking Threshold
if genSettings.FlagDMEdetectionOn
    dme_threshold = genSettings.dme_threshold;
end
%- End of Parameters of DME detection
%--------------------------------------------------------------------

% Here it is determined if the AGC shall is on and wether it shall mentain
% a constant signal power or it might only reduce the gain when the
% incomming signal power is so high that it might saturate the ADC
if genSettings.FlagAGCon & ~genSettings.Flag_use_look_up_table
    % use of PID controller
    if genSettings.FlagAGCfullRange
        h = waitbar (0.5, 'Computing Target Variance');
        gaussian_input = randn(1e6,1);
        var_gaussian_input = std(gaussian_input);
```

```
        gaussian_input_new = gaussian_input / var_gaussian_input * L/kADC;

        % Quantization of the Gaussian input
        [index, quant]= quantiz(gaussian_input_new,...
            ADC_thresholds, ADC_range);

        %--------------------------------------------------------------------
        % target Output variance
        var_target = std(quant);
        close (h);
    else
        var_target = var_normal;
    end;    %- if genSettings.FlagAGCfullRange


elseif genSettings.FlagAGCon & genSettings.Flag_use_look_up_table
    % use of look-up-table
    load(genSettings.look_up_table_path, 'gain_table', 'sigma_table')

end;  %- if genSettings.FlagAGCon & ~genSettings.Flag_use_look_up_table

fseek(inputfile(m),0,'bof');

clear signal signal_new sig_spec magnitude_sig_spec iterations_for_statistics
%--------------------------------------------------------------------
%- End of signal statistics computation
%--------------------------------------------------------------------


%--------------------------------------------------------------------
%--------------------------------------------------------------------
%Quantization, Frequency nulling and AGC gain adaption
%--------------------------------------------------------------------
%--------------------------------------------------------------------

h = waitbar (0, 'Computing quantized samples');
h_counter = 0;
epsilon = 0; % difference between target and present variance
gain_init = 10^(genSettings.AGC_gain_init/20);
gain = gain_init;

%- register for detection of temporal power fluctuations (DME)
```

```
shift_register = zeros(1,genSettings.length_shift_register);

%-------------------------------------------------------------------------
%- initializations for sliding hanning window
gain_old            = gain;
%- quantized noise
noise               = randn(1,round(sampleInterval));
noise_kADC          = noise / std(noise) * L_new / kADC;
[indx, quant]       = quantiz(noise_kADC, ADC_thresholds, ADC_range);

quant_cw_free       = quant .* window;

quant_store         = quant(round(sampleInterval/2) + 1:sampleInterval);
%-------------------------------------------------------------------------


% for test purpose: counter for histogram
p_block1 = zeros(Nant, nBlocks);   % percentage of spectral lines that
                                   % were above threshold
p_block2 = zeros(Nant, nBlocks);   % percentage of spectral lines that
                                   % were above threshold
p_block_dme = zeros(Nant, nBlocks); % how many values were above
                                    % detection threshold
z = 0;

for m = 1 : Nant

   for k = 1 : nBlocks

       h_counter = h_counter + 1;              % Waitbar
       waitbar(h_counter / (nBlocks * Nant));

       signal = gain_front_end * fread(inputfile(m), sampleInterval,...
           genSettings.outDataFormat);

       %- Signal level adjustenment
       signal_new = signal * gain;

       %- Quantization
       [indx, quant]= quantiz(signal_new, ADC_thresholds, ADC_range);

       % weight quant with current gain, so that spectral
```

```
% threshold stay fixed an no errors occur for overlapping windows
quant_a = quant * gain_init / gain;
%if max_soft_agc < (gain_init / gain);
%    quant_a = quant * gain_init / gain;
%else
%    quant_a = quant * max_soft_agc;
%end

%-----------------------------------------------------------------------
%- RFI DETECTION AND MITIGATION
%-----------------------------------------------------------------------
if flag_sliding_hann

    %-------------------------------------------------------------------
    %- processing on overlapping block
    quant_overlapping   = [quant_store, quant_a(...
        1:round( sampleInterval/2 ))];

    %- store second half of current block
    quant_store         = quant_a((round( sampleInterval/2 )...
        + 1):sampleInterval);

    %- wheigten with window
    quant_w1            = window .* quant_overlapping;

    %- frequency spectrum of overlapping block
    sig_spec1           = fft(quant_w1, genSettings.FFTpoints);
    magnitude_sig_spec1 = abs(sig_spec1);

    % Detection
    index1              = find(magnitude_sig_spec1 > ...
        spectra_detection_threshold);
    %- Frequency excision
    % Replace Spectra above threshold
    sig_spec1(index1)    = 0;

    quant_overlapping_cw_free = real(ifft(sig_spec1,...
        genSettings.FFTpoints));

    %- First Data output block without the effects of windowing
    quant_nomore_hann_first = quant_overlapping_cw_free(...
         1:round(sampleInterval/2)) + quant_cw_free(...
```

```
        (round( sampleInterval/2 ) + 1):sampleInterval);


    %-------------------------------------------------------------
    % Frequency Excision current block

    %- processing on current block
    %- window of this block
    quant_w2             = window .* quant_a;

    %- frequency spectrum of overlapping block
    sig_spec2            = fft(quant_w2, genSettings.FFTpoints);
    magnitude_sig_spec2  = abs(sig_spec2);

    % Detection
    index2               = find(magnitude_sig_spec2 >...
        spectra_detection_threshold);
    %- Frequency excision
    % Replace Spectra above threshold
    sig_spec2(index2)     = 0;

    quant_cw_free        = real(ifft(sig_spec2, genSettings.FFTpoints));

    %- undo the hanning windowing
    quant_nomore_hann_sec = quant_cw_free(1:round...
        ( sampleInterval/2 )) + quant_overlapping_cw_free...
        ((round( sampleInterval/2 ) + 1):sampleInterval);

    %- combine the two sets of data
    quant_rfi_free       = [quant_nomore_hann_first,...
        quant_nomore_hann_sec];


    %- for test purpose
    p_block1(m,k) = length(index1) / (2*(f_end-f_start));
    p_block2(m,k) = length(index2) / (2*(f_end-f_start));
    %-

else %- no sliding hanning window

    % Windowing
    quant_window     = window.*quant_a;
```

```
    %- Spectrum of quantized signal
    sig_spec         = fft(quant_window, genSettings.FFTpoints);

    %- Magnitude Spectrum
    magnitude_sig_spec = abs(sig_spec);

    %-------------------------------------------------------------------
    %- CW detection
    index            = find(magnitude_sig_spec > ...
        spectra_detection_threshold);
    %-------------------------------------------------------------------

    %- Frequency excision
    % Replace Spectra above threshold
    sig_spec(index)  = 0;

    quant_cw_free    = real(ifft(sig_spec, genSettings.FFTpoints));

    quant_rfi_free   = quant_cw_free;


    %- for test purpose
    p_block1(m,k) = length(index) / (2*(f_end-f_start));
    %-
end; %- if sliding window



%-------------------------------------------------------------------
%- END OF RFI DETECTION AND MITIGATION
%-------------------------------------------------------------------


%-------------------------------------------------------------------
%- Gain adjustment
%-------------------------------------------------------------------
if genSettings.FlagAGCon & genSettings.FlagDMEdetectionOn
    %- The Gain needs to be adjusted to the incomming signal power
    %  for the DME free case
    tmp               = [shift_register quant];
    power_in_register = zeros(1,sampleInterval);
```

```
for ii = 1:(sampleInterval)
    power_in_register(ii)    = std(tmp(ii:(ii + ...
        genSettings.length_shift_register))).^2;
end;

%above_dme_threshold = find(power_in_register > dme_threshold);

% also kick those values right before and right after the
% dme threshold viloation
dme_start    = 1;
dme_end      = 1;
kick         = [];

while (dme_start ~= sampleInterval) & (dme_end ~= ...
      sampleInterval) & ~isempty(dme_start) & ~isempty(dme_end)

     dme_start       = find(power_in_register(dme_end:...
         sampleInterval) > dme_threshold,1) + dme_end - 1;
     dme_end         = find(power_in_register(dme_start:...
         sampleInterval) < dme_threshold,1) + dme_start - 1;

     if ~isempty(dme_start) & isempty(dme_end)
         dme_end     = length(tmp);
     end;

     %- increase blanking range by

     dme_start_pos   = dme_start_pos -...
         genSettings.increase_blanking_range;
     dme_end_pos     = dme_end_pos +...
         genSettings.increase_blanking_range;
     kick            = [kick dme_start_pos:dme_end_pos];
end;

% take care that there are no postion marks exceed the range of
% quant
kick    = kick(find(kick >= genSettings.length_shift_register));
kick    = kick(find(kick <= sampleInterval));

quant_to_AGC   = quant;
% Blanking
```

```
    quant_to_AGC(kick) = 0;

    quant_to_AGC   = quant_to_AGC( find(abs(quant_to_AGC > 0)) );

    %- init register for next run
    shift_register     = quant( (sampleInterval - ...
        genSettings.length_shift_register + 1):sampleInterval );

else
    quant_to_AGC       = quant;

end   %- if genSettings.FlagAGCon & genSettings.FlagDMEdetectionOn


if genSettings.FlagAGCon & ~genSettings.Flag_use_look_up_table
    %- AGC with PID controller
    %- save old gain
    gain_old     = gain;

    %- Difference between target and present power
    epsilon      = std(quant_to_AGC)^2 - var_target^2;

    %- Set Gain
    gain         = gain + gain / gain_init *...
         genSettings.AGC_Ki * epsilon;

    % The gain of the amplifier cannot become smaller than 0dB
    if gain < 1
        gain = 1;
    end;

    % The gain of the amplifier cannot become larber than 20dB
    if gain > gain_init
        gain = gain_init;
    end;

elseif  genSettings.FlagAGCon & genSettings.Flag_use_look_up_table
    %- AGC with look-up-table
    gain_old     = gain;

    % table-search algorithm
    % estimate correct gain on basis of output variance
```

```
    a = find(std(quant_to_AGC) > sigma_table(gain_pos,:),1,'last');
    b = find(std(quant_to_AGC) < sigma_table(gain_pos,:),1,'first');

    if isempty(a)
        gain_pos = b;
    elseif isempty(b)
        gain_pos = a;
    else
         if (abs(std(quant_to_AGC) - sigma_table(a,:)))...
                 > (abs(std(quant_to_AGC) - sigma_table(b,:)))
             gain_pos = b;
         else
             gain_pos = a;
         end
    end

    gain = gain_table(gain_pos);

end; %- if
%-----------------------------------------------------------------------
%- End of gain adjustment
%-----------------------------------------------------------------------


%-----------------------------------------------------------------------
%- Reduction of Qunatization range for further signal processing
%- Force the now interference free Signal into the resolution range
%  and write the data to an output file

%adjust the signal power for minimum quantization losses
data = quant_rfi_free / std(quant_rfi_free)...
    * abs(soft_quant_thresholds(1)) / kADC_soft;
[ind, output]= quantiz(data, soft_quant_thresholds, soft_quant_range);


count = fwrite(outputfile(n), output, genSettings.outDataFormatADC);
if (count ~=  sampleInterval),
    error('something wrong with writing the data file!');
    fclose(outputfile(n));
end;
```

```
       %-for test purpose
       %qq(:,k) = quant;
       ggain(m,k)= gain;
       gepsilon(m,k) = epsilon;
       if mod(k,100) == 0
           z = z+1;
           histin(:,z) = hist(quant,ADC_range);
           histbetween(:,z) = hist(quant_cw_free,ADC_range);
           histout(:,z) = hist(output,soft_quant_range);
       end;

   end;


end; % for m = 1 : Nant


fclose all;
close(h);
clear h_counter;

%- for test purpose
genResults.p_block1 = p_block1; % Percentage of detection in overlapped
                                % window

genResults.p_block2 = p_block2; % Percentage of detection in nonoverlapped
                                % window
genResults.p_block_dme = p_block_dme; % percetnateg of DME threshold
                                      % violations
genResults.histin = histin;     % Histogram after ADC
genResults.histout = histout;   % Histogram after Software ADC
genResults.histbetween = histbetween; % Histogram after Frequency Excision
genResults.gain = ggain;        % Gain
%---

genResults.sDataFileADC = sDataFileADC; % output filename

end
```

A.5. Blanking Techniques

The program code shown here describes the circuit presented in Figure 6.2. In the "Settings"-File one can decide whether to perform blanking on a sample-by-sample basis or by monitoring power fluctuations in the received signal.

A.5.1. Settings

```
function genSettings = generatorSettings
% Setting the GNSS generator parameters
%
% Usage:
%   genSettings = generatorSettings
%
% Inputs: no inputs needed
%
% Outputs:
%   genSettings = structure containing generator setting parameters
%

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%Thorsten

genSettings.Nant            = 1;
genSettings.fIF             = 4e6;        % intermediate frequency [Hz]
genSettings.fs              = 2e7;        % sampling frequency [Hz]
genSettings.sampleInterval  = 2e4;


genSettings.AGC_gain_init   = 20;         % [dB] AGC gain at it's start

genSettings.nBitsADC = [6 8];
            % [a b] a = number of bits used by ADC for signal resolution
            %       b = dynamic range of ADC for interference mitigation

genSettings.look_up_table_path = 'Data\look_up_table_68.mat';

genSettings.nBits_SoftQuant = 6;     % number of bits for correlator
genSettings.L = 1;     % Maximum quantization Threshold of A/D converter
```

```
genSettings.get_statisticcs_time    = 0.1;
            % time in which statistics shall be computed
genSettings.outDataFormatADC        = 'int8';
            % Data Format after the ADC

% Blanking
genSettings.sample_by_sample = 0;
            % blanking on sample-by-sample = (1)
            % blanking on power fluctuations in the
            % received signal = (0)
genSettings.length_shift_register   = 20;

genSettings.increase_blanking_range  = 15;
            % Number of samples which will also be blanked
            % before and after DME threshold violation, respectively

%--- Output data format
genSettings.outDataType     = 'real';
            % possible options: {'real', 'I&Q'}
genSettings.outDataFormat   = 'double';
            % possible options: {'int8', 'schar', 'double'}
end
```

A.5.2. Program Code

```
function genResults = blanking(genSettings, genResults)
% Quantization and pre-correlation Interference Mitigation of the generated
% GNSS signal
%
% Usage:
%       blanking(genSettings, genResults)
%
% Inputs:
%   genSettings = structure containing generator setting parameters
%   genResults  = structure containing parameters of the generated signal
%
% Usage:
%   [flagOK, genResults] = produceSignal(genSettings)
```

```
%
% Inputs:
%   genSettings = structure containing generator setting parameters
%
% Outputs:
%       a file containing the 'digitized data'


% Thorsten Lotz, Diploma Thesis at DLR

%--------------------------------------------------------------------------
%--------------------------------------------------------------------------

%Parameters
% number of antenna elements
Nant = genSettings.Nant;

% number of elements used for interference detection and AGC setting
sampleInterval = genSettings.sampleInterval;


fs = genSettings.fs;


%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%--- Open the files obtained from the GNSS signal generator
for m = 1 : Nant


    %Open the files obtained from the GNSS signal generator
    [inputfile(m), message] = fopen(genResults.sDataFile{m}, 'r');


    if (inputfile(m) < 0)
        error([' Error opening the data file ', genResults.sDataFile{m}])

    else
        fseek(inputfile(m), 0, 'bof');       %Set pointer to first element

        %data obtains the incomming signal
        [signal, count] = fread(inputfile(m), sampleInterval,...
```

```
            genSettings.outDataFormat);

        if (count < sampleInterval)
        % The file is too short
        error('Could not read enough data from the data file');
        end;

    end;
end;



%--- Assign names for output data file
for n = 1 : Nant
    sDataFileADC{n} = ['adc_blanking_',genResults.sDataFile{n}((find(...
      genResults.sDataFile{n} == '\')+1):length(genResults.sDataFile{n}))];
    outputfile(n) = fopen(['Results\', sDataFileADC{n}], 'w');
end



%-- determine some loop parameters
switch genSettings.outDataFormat
    case 'int8'
        data_type_size = 1;
    case 'schar'
        data_type_size = 1;
    case 'double'
        data_type_size = 8;
    otherwise
        data_type_size = 1;
end


% determine the total number of Samples in the File
fseek(inputfile(1), 0, 'eof');
total_number_of_samples = ftell(inputfile(1)) / data_type_size;
fseek(inputfile(1), 0, 'bof');


% number of blocks for the for loop
nBlocks = floor(total_number_of_samples / sampleInterval);
```

```
% rest of the Samples in the file, not quantized in the loop
sampleRest = floor( mod(total_number_of_samples, sampleInterval) );


%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Compute the ADC parameters

% =^ highest quantization threshold, e.g. ADC range is between -1V and +1V
L = genSettings.L;

% the ADC output bins are integer values
maxADCBin = 2^(genSettings.nBitsADC(1) + genSettings.nBitsADC(2))-1;

% number of total ADC bins
nLevelsADC = 2^(genSettings.nBitsADC(1) + genSettings.nBitsADC(2));

% ADC quantization thresholds
ADC_thresholds = linspace(-L, L, nLevelsADC-1);

% ADC output range (ADC bins)
ADC_range = linspace(-maxADCBin, maxADCBin, nLevelsADC);


%--- determine ADC resolution range factor [Bastide Ph.D thesis, p. 63]
switch genSettings.nBitsADC(1)
   case {1, 2}
       kADC = 0.986;
   case 3
       kADC = 1.731;
   case 4
       kADC = 2.291;
   case 5
       kADC = 2.7225;
   case 8
       kADC = 3.0;
   otherwise
       kADC = 3.0;
end

%--- determine optimal ratio factor for software quantization [Bastide Ph.D thesis,
switch genSettings.nBits_SoftQuant
```

```
   case {1, 2}
       kADC_soft = 0.986;
   case 3
       kADC_soft = 1.731;
   case 4
       kADC_soft = 2.291;
   case 5
       kADC_soft = 2.7225;
   case 8
       kADC_soft = 3.0;
   otherwise
       kADC_soft = 3.0;
end

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% find the maximum threshold for the resolution bins
% see Basttide p. 171 ff.;
% 'abs(ADC_thresholds(1)-ADC_thresholds(2))' is the quatization step size
L_new = (2^(genSettings.nBitsADC(1) - 1) - 1) *...
    abs(ADC_thresholds(1)-ADC_thresholds(2));

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% in order to reduce the total range of the ADC to only resolution range
% one needs to 'quantize' one more time, therefore compute some parameters
% of the second software quantization, the data output of the quantizer is
% given to the correlator
soft_quant_thresholds = linspace(-(2^genSettings.nBits_SoftQuant - 1),...
    2^genSettings.nBits_SoftQuant - 1, 2^genSettings.nBits_SoftQuant-1);
soft_quant_range = linspace(-(2^genSettings.nBits_SoftQuant - 1),...
    2^genSettings.nBits_SoftQuant - 1, 2^genSettings.nBits_SoftQuant);
%-------------------------------------------------------------------------

clear index quant

%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% Get some Statistics of the Interference free Signal
% It is assumed that within the first genSettings.get_statisticcs_time sec
% no Interference is present
%-------------------------------------------------------------------------
```

```
iterations_for_statistics = floor(genSettings.get_statisticcs_time * fs...
    / sampleInterval);

h = waitbar (0, 'Computing signal statistics');
h_counter = 0;


%- Standart deviation of the undistored signal
std_signal_norm      = zeros(1, iterations_for_statistics);

%- Target variance of quantized signal
var_target      = zeros(1, iterations_for_statistics);

for iter = 1:iterations_for_statistics

    h_counter           = h_counter + 1;              % Waitbar
    waitbar(h_counter / iterations_for_statistics);

    [signal, count]     = fread(inputfile(m), sampleInterval,...
        genSettings.outDataFormat);

    std_signal_norm(iter)    = std(signal);

    % Power adjustenment of the incomming signal according to the ADC range
    % only within the first 100 ms std(signal) is assumed to be known!
    signal_new          = signal / std(signal) * L_new / kADC;

    % Quantization of the test statistic
    [index, quant]      = quantiz(signal_new, ADC_thresholds, ADC_range);

    % Target variance for AGC setting and Blanking
    var_target(iter)    = std(quant);
end;

close (h);
fseek(inputfile(m),0,'bof');

%-------------------------------------------------------------------
%- Set Front-End Gain
% It is assumed that in the first 100 ms of the Signal no interference is
% present and one knows the standart deviation of the incomming signal.
% So the total gain of the receivers front end is set according to kADC and
```

```
% genSettings.AGC_gain_init
gain_front_end  = L_new / kADC / 10^(genSettings.AGC_gain_init/20)...
    / (sum(std_signal_norm) / iterations_for_statistics);

%- Target variance for AGC setting and Blanking
var_target      = sum(var_target) / iterations_for_statistics;

%- Variance of Signal before ADC
std_signal_norm = sum(std_signal_norm) / iterations_for_statistics;


%-------------------------------------------------------------------------
%- Blanking Threshold
if genSettings.sample_by_sample
    dme_threshold   = 2^genSettings.nBitsADC(1) - 1;
else
    dme_threshold   = (2^genSettings.nBitsADC(1) - 1)^2...
        * genSettings.length_shift_register;
end
%-------------------------------------------------------------------------
%- End of signal statistics computation
%-------------------------------------------------------------------------


%-------------------------------------------------------------------------
%- Quantization and Interference mitigation
%-------------------------------------------------------------------------

h           = waitbar (0, 'Computing quantized samples');
h_counter   = 0;

%- Register to save current position in look-up-table
gain_pos        = 1;
gain_init = 10^(genSettings.AGC_gain_init/20);
gain = gain_init;

%- register for detection of temporal power fluctuations
shift_register = zeros(1,genSettings.length_shift_register);

%- load look-up-table
load(genSettings.look_up_table_path, 'gain_table', 'sigma_table')
```

```
% for test purpose:
% how many values were above detection threshold
p_block_dme = zeros(Nant, nBlocks);

z = 0;

for m = 1 : Nant

   for k = 1 : nBlocks

       h_counter        = h_counter + 1;              % Waitbar
       waitbar(h_counter / (nBlocks * Nant));

       signal           = gain_front_end * fread(inputfile(m),...
           sampleInterval, genSettings.outDataFormat);

       %- Signal level adjustenment
       signal_new       = signal * gain;

       %- Quantization
       [indx, quant]    = quantiz(signal_new, ADC_thresholds, ADC_range);

       %--------------------------------------------------------------------
       %- RFI DETECTION AND MITIGATION
       %--------------------------------------------------------------------

       %- Blanking
       if genSettings.sample_by_sample

        kick = find(abs(quant) > dme_threshold);

       else
        tmp               = [shift_register quant];
        power_in_register = zeros(1,sampleInterval);

        for ii = 1:(sampleInterval)
           power_in_register(ii)    = std(tmp(ii:(ii +...
               genSettings.length_shift_register))).^2;
        end;

        % also kick those values right before and right after the dme threshold vilo
        dme_start    = 1;
```

```
 dme_end      = 1;
 kick         = [];

 while (dme_start ~= sampleInterval) & (dme_end ~= sampleInterval)...
        & ~isempty(dme_start) & ~isempty(dme_end)

     dme_start       = find(power_in_register(dme_end:...
         sampleInterval) > dme_threshold,1) + dme_end - 1;
     dme_end         = find(power_in_register(dme_start:...
         sampleInterval) < dme_threshold,1) + dme_start - 1;

     if ~isempty(dme_start) & isempty(dme_end)
         dme_end     = length(tmp);
     end;

     %- increase blanking range by

     dme_start_pos   = dme_start_pos -...
         genSettings.increase_blanking_range;
     dme_end_pos     = dme_end_pos +...
         genSettings.increase_blanking_range;
     kick            = [kick dme_start_pos:dme_end_pos];
 end;

 % take care that there are no postion marks exceed the range of quant
 kick         = kick(find(kick >= genSettings.length_shift_register));
 kick         = kick(find(kick <= sampleInterval));
end;

% Blanking
 quant_dme_free = quant;
 quant_dme_free(kick) = 0;

%-------------------------------------------------------------------
% Adjust Gain
%-------------------------------------------------------------------
% table-search algorithm
% estimate correct gain on basis of output variance
std_to_agc = std(quant_dme_free (find (abs (quant_dme_free)>0) ) );

a = find(std_to_agc > sigma_table(gain_pos,:),1,'last');
b = find(std_to_agc < sigma_table(gain_pos,:),1,'first');
```

```
if isempty(a)
     gain_pos = b;
elseif isempty(b)
     gain_pos = a;
else
     if (abs(std_to_agc - sigma_table(a,:))) >...
             (abs(std_to_agc - sigma_table(b,:)))
         gain_pos = b;
     else
         gain_pos = a;
     end
end
gain = gain_table(gain_pos);
%--------------------------------------------------------------------
%- End of gain adjustment
%--------------------------------------------------------------------

quant_rfi_free = quant_dme_free;
%--------------------------------------------------------------------
%- Reduction of Qunatization range for rurther processing
%- Force the now interference free Signal into the resolution range
%  and write the data to an output file

%adjust the signal power for minimum quantization losses
if genSettings.nBitsADC(1) == genSettings.nBits_SoftQuant
    data_new     = quant_rfi_free;
else
    data_new     = quant_rfi_free / std(quant_rfi_free)...
        * abs(soft_quant_thresholds(1)) / kADC_soft;
end;
[ind, output]= quantiz(data_new, soft_quant_thresholds,...
    soft_quant_range);


count        = fwrite(outputfile(n), output,...
    genSettings.outDataFormatADC);
if (count ~=  sampleInterval),
    error('something wrong with writing the data file!');
    fclose(outputfile(n));
end;
```

```
        %- for test purpose
        p_block_dme(m,k) = length(kick) / sampleInterval;
        %-

        %-for test purpose
        %qq(:,k) = quant;
        ggain(m,k)   = gain;

    end;

end; % for m = 1 : Nant


fclose all;
close(h);
clear h_counter;

%- for test purpose
genResults.p_block_dme = p_block_dme;
genResults.gain = ggain;
%---

genResults.sDataFileADC = sDataFileADC;
end
```

B. Bibliography

[Amo83] Amoroso, Frank: *Adaptive A/D Converter to Suppress CW Interference in DSPN Spread-Spectrum Communications.* IEEE TRANSACTIONS ON COMMUNICATIONS, Oktober 1983.

[Bal07] Balaei, Asghar Tabatabaei: *A preventative approach to mitigating CW interference in GPS receivers.* Springer, University of New South Wales, 2007.

[Bas03] Bastide, Frederic: *Automatic Gain Control (AGC) as an Interference Assessment Tool.* Institute of Navigation: GPS/GNSS and NTM, September 2003.

[Bas04] Bastide, Frédéric: *Analysis of the Feasibility and Interests of Galileo E5a/E5b and GPS L5 Signals for Use with Civil Aviation.* Doctor Thesis, Touluse, 2004.

[Bor07] Borre, Kai: *A Software-Defined GPS and Galileo Receiver.* Birkhäuser, Boston, 2007.

[BWP96] Bradford W. Parkinson, James J. Spilker Jr.: *Global Positioning System: Theory and Applications, Volume 1.* American Institute of Aeronautics and Astronautics, Inc, SW, Washington, 1996.

[Cho91] Cho, K.M.: *Optimum gain control for A/D conversion using digitized I/Q data inquadrature sampling.* IEEE Transactions on Aerospace and Electronic Systems, Jan 1991.

[Des04] Deshpande, Sameet Mangesh: *Global Positioning System: Signals, Measurements and Performance.* Department of Geomatics Engineering, University of Calgary, 2004.

[Die00] Dierendonck, A.J. Van: *GPS Signal Structure.* 2000. http://www.ima.umn.edu/talks/workshops/8-16-18.2000/van-

dierendonck/.

[Eng06] Enge, Per: *Global Positioning System: Signals, Measurements and Performance*. Ganga-Jamuna Press, Lincoln, Massachusetts, 2006.

[Gra02] Grabowski, Joseph: *Characterization of L5 Receiver Performance Using Digital Pulse Blanking*. Proceeding of The Institute of Navigation GPS Meeting, Portland, 2002.

[Kap96] Kaplan, Elliot D.: *Understanding: GPS Principles and Applications*. Artech House, Norwood, 1996.

[Kon08] Konovaltsev, Andriy: *Local Integrity Navigation Augmentation: Technical Note 1*. DLR - Institute of Communications and Navigation, Oberpfaffenhofen, 2008.

[Lyo01] Lyons, Richard G.: *Understanding Digital Signal Processing*. Prentice Hall PTR, Upper Saddle River, 2001.

[Mar04] Marti, Lukas: *Interference Detection by Means of the Software Defined Radio*. ION GNSS 17th International Technical Meeting of the Satellite Division, 21-24 Sept. 2004.

[MR06] Mathieu Raimondi, Frederic Bastide: *Mitigating Pulsed Interference Using Frequency Domain Adaptive Filtering*. ION GNSS 19th International Technical Meeting of the Satellite Division, Fort Worth, TX, 26-29 Sept. 2006.

[Opp99] Oppenheim, Alan V.: *Discrete-Time Signal Processing*. Prentice Hall, New Jersey, 1999.

[Res08] Research, Interstellar: *Data AcQuisition And Real-Time Analysis*, 2008.
http://www.daqarta.com/eex06.htm.

[REZ95] Rodger E. Ziemer, Roger L. Peterson, David E. Borth: *Introduction to Spread Spectrum Communications*. Prentice Hall, Upper Saddle River, 1995.

[Rup00] Rupprecht, Werner: *Signale und Übertragungssysteme*. Fachbereit Elektro- und Informatinonstechnik, TU Kaiserslautern, 2000.

[Spi77] Spilker, James J.: *Digital Communications by Satellite*. Prentice Hall, February 1977.

[Ste79] Stearns, Samuel D.: *Digitale Verarbeitung analoger Signale*. Oldenbourg, Munich, 1979.

[Tsu00] Tsui, James Bao-Yen: *Fundamentals of Global Postitioninng System Receivers - A Software Approach*. John Wiley & Sons, Inc., New York, 2000.

[Web08a] Weber, Christian: *Galileo - Advanced Applications*. DLR, Oberpfaffenhofen, 2008.

[Web08b] Weber, Christian: *Interference Detection and Mitigation*, 2008.

[Wik] Wikipedia: *GLONASS*. http://en.wikipedia.org/wiki/GLONASS/.