Universität Stuttgart

Geodätisches Institut

Noise performance of the modernized GPS- and GALILEO-Systems

Studienarbeit im Studiengang

Geodäsie und Geoinformatik

an der Universität Stuttgart

Sebastian Gerst

Stuttgart, September 2010

Betreuer und Prüfer: Prof. Dr. sc. techn. Wolfgang Keller
Universität Stuttgart

Erklärung der Urheberschaft

Hiermit erkläre ich, Sebastian Gerst, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum Unterschrift

Abstract

Mit der Modernisierung von GPS und dem Aufbau des neuen Europäischen Satellitensystems Galileo stehen neue Signale zur Verfügung, deren Rauschverhalten bisher nur ungenügend verstanden wird. Die neuen Signale werden zur Zeit noch nicht operationell eingesetzt, sind aber im Falle von GPS mittlerweile in zwei Satelliten implementiert, für Galileo sind zwei Testsatelliten im Orbit, die ebenfalls neue Signale aussenden. Aufgezeichnet wurden die neuen, als auch die herkömmlichen Signale mit Hilfe eines Septentrio Empfängers.

Das Ziel der Arbeit ist die Anteile von dem empfangenen Signal abzutrennen, die von dem zeitlichen Verlauf der Messung abhängen. Dies ist zum einen die Satellit-Empfänger-Geometrie und zum anderen der ionosphärische Laufzeitfehler. Der verbleibende Signalanteil ist dann als Rauschen aufzufassen, dessen Korrelationsverhalten und Leistungsspektrum diskutiert werden sollen.

Schlagwörter

- GNSS
- GPS
- Galileo
- Rauschverhalten
- Korrelationsverhalten
- Septentrio

Contents

List of Figures

List of Tables

1 Introduction

Global Navigation Satellite Systems provide location and time information at all times anywhere on or near Earth. For GPS[1], now operating since more than 30 years, a modernization process has started to provide more performance and to be competitive with other systems like the new Galileo[2] or GLONASS[3]. With this modernization process, new signals like GPS (L5) or Galileo (L1, E5a, E5b, E6) are applied to the system. Most of the systems provide in future 3 or 4 signals, one or two more, than in the past. These signals are also provided for civilian usage and they are in protected frequency bands, so they can be used for human safety applications like aircraft landing systems.

Generally all GNSS[4] are affected by error sources, which decrease the positioning accuracy. Some of these error sources can be modeled well, like tropospheric refraction, other's can only be modeled worse, like ionospheric refraction and some of them can't be modeled, because they have random characteristics like thermal noise. To improve the positioning accuracy, filter techniques can be applied. But to set up filters, it is recommended to know the behavior of the signal, which has to be used.

The main goal of this thesis is to understand the noise behavior of these new signals, to discuss the correlation behavior and the power spectrum.

1.1 Assignment of Tasks

First of all it is necessary to convert the received data, stored in a receiver specific format to an ASCII[5] format, for further use. To do this, Thales GmbH provides software and instructions.
After that, two different strategies are planned to extract the signal noise from the measurement.

The first strategy will be to estimate a polynomial function to the measurement, which will approximate the changing receiver - satellite position (geometry component). After a substraction of the estimated geometrical component, the noise should be left. Based on the fact, that changing geometry and ionospheric refraction are longwave effects and noise is high frequency, an extraction of noise should be possible with this strategy.

The second strategy is to use model-based frequency combinations to extract the ionospheric refraction from the measurement, to reduce the geometrical component of the ionospheric refraction left and to get the noise in the signal.

For both methods it is necessary to check if the estimations still have longwave components, this can be done by inspection of autocorrelation functions and spectral power densities. If there are indications of incomplete elimination of trends, the noise extraction strategies have to be modified.

If all strategies are consistent, it is possible to estimate continuous model functions into the data, of which the parameters of the correlation function and the power spectrum can be compared and discussed.

1.2 Thesis structure

Beginning with a general theoretical section, in which the different GNSS, the observation equations, the error sources of these systems and the general context of stochastic processes are discussed, will be followed by a section in which the signal conversion, from the receiver's raw data, to data, usable in Matlab, will be described.

Having extracted the measurement, it is recommended to extract the noise in the signal by using different

[1] GPS: Global Positioning System, American System
[2] Galileo: Named after Galileo Galilei (1564-1642), European System
[3] GLONASS: abbreviation of GLObal'naya NAvigationnaya Sputnikovaya Sistema, Russian System
[4] GNSS: Global Navigation Satellite System
[5] ASCII: American Standard Code for Information Interchange

approaches, which will be done in chapter four.

The last chapter includes the analysis and discussion of the noise behavior.

The appendix will include the source code and it's description.

2 GNSS Basics

This chapter provides an overview of Global Navigation Satellite Systems. It deals with the history of GNSS, the different systems and their specifications available today, their way of operation and at least the basics needed to deal with these systems.

After a short historical overview, the different specifications of the systems, an overview of satellite orbits, observation equations of GNSS and their error budget will be discussed.

2.1 History of GNSS

Before any GNSS was deployed, angular measurements to natural stars were the best method to measure ones position on earth (see [14]). With the development of radio signals in the 1960's the U.S. Navy introduced a navigation system, known as Transit, based on this kind of signal, measuring the Doppler shift as positioning information, which opened a new area of positioning precision, approx. 500 to 15m, depending on the receiver.
In 1973 the U.S. Department of Defence decided to develop a new satellite navigation system based on radio ranging which should increase ranging accuracy up to m-precision for military use. In fact, they tried to navigate 10 bombs into the same hole. Instead of angular measurements to stars, ranging measurements to artificial NAVSTAR[6]'s were performed. About 20 years later NAVSTAR GPS replaced Transit and a new era of navigation and point positioning began. GPS was fully operational in the 1990's.

Due to the cold war, the Russian federation developed nearly in the same time an equivalent system, called GLONASS. They began developing the system in 1972 and on September 24th 1993 it was explained as operational.

On May 27th 2003 the European Union decided to develop an own navigational satellite system Galileo, with the difference of civilian use in contrast to GPS and GLONASS which are operated by national military. Nowadays, Galileo is still in an expansion process, two test satellites, GIOVE-A and GIOVE-B are in orbit.

For a few years, a renewal process was started, which will improve performance and availability for civilian use of GNSS. In the case of GPS, new satellites are developed and placed in orbit. Modern GNSS provide at least three frequencies in comparison to the old configuration of GPS, which had two frequencies.

2.2 Satellite orbits

Analyzing the astronomical observations of Tycho Brahe[7], Kepler[8] was the first to give a mathematical description to (planetary) orbits. He published his first two laws in 1609 in his Astronomia Nova. His third law was published ten years later, 1619 in Harmonice Mundi (for more information, see Sneeuw (2006, [17])).

A Kepler orbit is defined by six parameters.

[6] NAVSTAR: Navigational Satellite Time and Ranging
[7] Tycho Brahe (1546-1601). A Danish nobleman, known for his accurate and comprehensive astronomical and planetary observations.
[8] Johannes Kepler (1571-1630). Born in Weil der Stadt and studied at Tübingen University.

Table 2.1: Kepler elements

Parameter	Description
a	Semi-major axis
e	Eccentricity
I	Inclination
Ω	Right ascension of ascending node
ω	Argument of perigee
ν	True anomaly

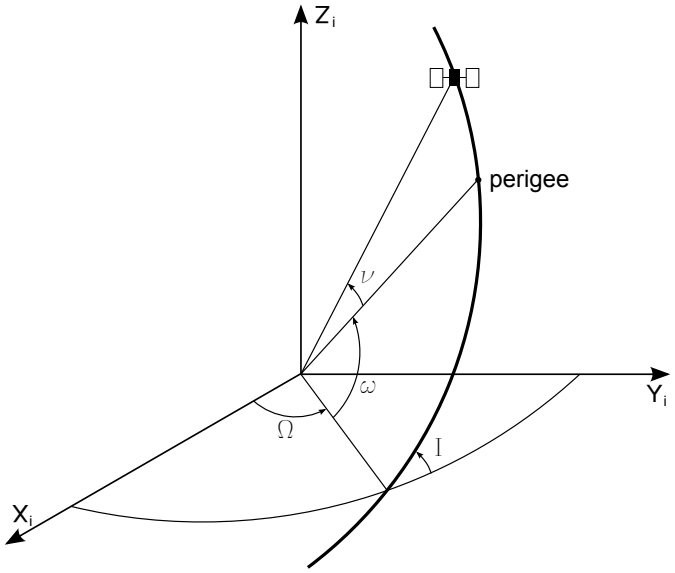


Figure 2.1: Geometry of the Kepler orbit in the inertial System

2.2.1 First law

Kepler's first law states, that planets move on conic sections around the sun. Only if these conic sections are closed, the shape of this sections represent an ellipse. Or a satellite moves in ellipses around a planet, located in the focus of an ellipse, being the center of mass.

The polar equation of an ellipse is

$$r(\nu) = \frac{a(1-e^2)}{1+e\cos\nu} \tag{2.1}$$

where r is the distance between one of the focal points and a point on the ellipse.

2.2.2 Second law

The second law is called area law. It describes that the line between focal point and satellite or a planet covers in equal intervals Δt equal areas A. The consequence of that law, the angular velocity $\dot{\nu}$ must be varying over time. It reaches the maximum at perigee and the minimum at apogee.

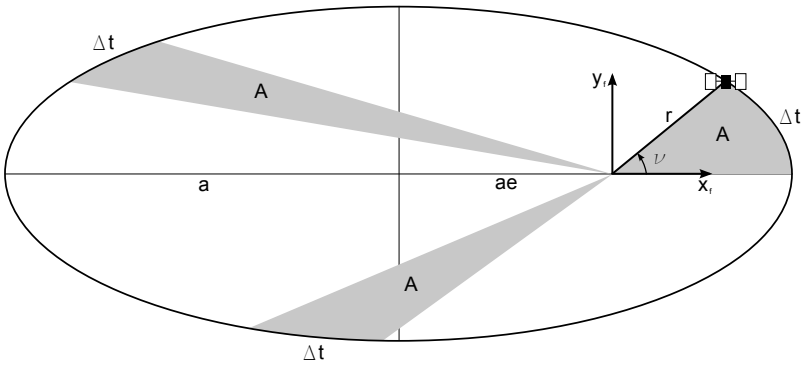


Figure 2.2: Geometry of an ellipse with perifocal coordinate system

2.2.3 Third law

"The square of the orbital period of a planet is directly proportional to the cube of the semi-major axis of its orbit."

$$n^2 a^3 = GM \tag{2.2}$$

where $n = \frac{2\pi}{T}$.

2.3 Principles of Operation

2.3.1 Signal

GNSS signals are L-Band signals. This means, they are operating at a frequency band of 1.5 GHz. To generate the signals, sent by a satellite, a fundamental frequency f_0 is generated by an oscillator with an atomic clock. To get the nominal carrier frequency f_c the fundamental frequency f_0 is multiplied by an integer value. So one is able to generate different carrier frequencies out of one fundamental one. To modulate navigation data on to the signal, different modulation types like Binary Phase Shift Keying (BPSK) or Binary Offset Carrier (BOC) are applied.

2.3.2 Modulation types

To modulate digital binary data like navigation data to the carrier, different procedures are implemented in the GNSS.

Binary Phase Shift Keying (BPSK) is a technique to represent digital data on a reference signal. Phase Shift Keying (PSK) changes the phase of the signal to modulate data on it and Binary Phase Shift Keying is the simplest form of PSK where only two phases, separated by 180° are used (see [8] and [7]).

Binary Offset Carrier (BOC) modulation uses a sub-carrier frequency, equal or higher to the chip rate, which is multiplied to the signal (see [2]).

Alternative Binary Offset Carrier (AltBOC) is a special case of BOC modulation (see [2]).

2.3.3 Positioning

The fundamental positioning principle of GNSS is the use of one way ranging signals. GNSS satellites are broadcasting their estimated positions in space. Measurements are the ranges between satellite and receiver, which will be done by correlating the received signal with a replica signal generated in the receiver. With this correlation, one is able to determine the run-time of the signal between satellite and receiver. Multiplied with the speed of light we get the distance. To estimate the user position,

measurements to at least four satellites at the same epoch are required. Three for the unknown three dimensional position of the user and one to solve the unknown receiver clock error (see chapter 2.5.3).

2.4 System specifications

All GNSS are based on the ranging measurement principle for point positioning. In fact, each system is a little bit different to each other, but all systems are using radio signals and they are located at approx. the same orbital height and at the same inclination.

GNSS are separated in three segments, the

Space segment Which consists of the satellites itself.

Ground segment Which consists of ground stations tracking satellites, determining the satellite's orbital parameters, providing information about the system and operating it.

User segment Which consists mainly of civilian users, positioning themselves with ranging signals. Other applications can be military or scientific, respectively GNSS could also be used as a source of precise time information.

In the following, the system specifications of each system will be discussed.

2.4.1 GPS

Global Positioning System was the first fully operational and stable satellite navigation system, maintained and operated by the U.S. Air Force.
Today most applications available are based on GPS measurements. System specifications are referring to [4].

Space segment

With it's history, now over 35 years, the GPS system was originally designed with 24 satellites and had several generations of satellite types, which were modernized over time.

Table 2.2: Satellite launches

Block	Launch Period	Satellite launches	Currently in orbit and healthy
1	1978 - 1985	11	0
2	1989 - 1990	9	0
2A	1990 - 1997	19	10
2R	1990 - 1997	13	12
2R-M	2005 - 2009	8	7
2F	2010 - 2011	1	0
3A	2014 -	0	0

If we sum up the satellites currently in orbit and healthy we get a number of 29 satellites. At this time, PRN01 from block 2R-M, which is transmitting the new L5 signal is set to unhealthy.

The layout of the satellite orbits has six orbital planes, with four satellites on each. These six planes have an inclination of $I = 55°$ and are separated by 60° right ascension of the ascending node.

Table 2.3: Orbital parameters of GPS

Parameter	Value	Description
	29	Number of satellites
	6	Orbital planes
	~ 7	Number of satellites per orbital plane
$\Delta\Omega$	60°	Separation of ascending node per orbital plane
I_0	55°	Nominal inclination
a	26561 km	Semi-major axis
T	11h 58min	Revolution time

GPS satellites are transmitting the following signals

Table 2.4: GPS frequencies

Frequency [MHz]	Name	Modulation Type	Usage
1575.42	L1	C/A-Code BPSK(1)	civilian
1575.42	L1	P(Y)-Code	military
1227.60	L2	P(Y)-Code	military
1227.60	L2C	BPSK(1)	civilian
1176.45	L5	BPSK(10)	civilian

Control segment

The control segment is composed of

- a master control station (Shriever Air Force Base, Colorado Springs)
- an alternate master control station
- four dedicated ground antennas
- six dedicated monitoring stations

2.4.2 Galileo

Galileo is the GNSS currently being built by the European Union and the European Space Agency. In contrast to GPS, Galileo will be under civilian control. Today two test satellites, GIOVE-A and GIOVE-B are in orbit. The whole system should be operational in 2014. System specifications are referring to [3].

Space segment

The future space segment of Galileo will consist of 30 satellites in a Walker constellation, on 3 orbital planes, resulting in ascending nodes separated by 120° longitude.

Table 2.5: Orbital parameters of Galileo

Parameter	Value	Description
	30	Number of satellites
	3	Orbital planes
	9 + 1	Number of satellites per orbital plane + one backup
$\Delta\Omega$	120°	Separation of ascending node per orbital plane
I_0	56°	Nominal inclination
a	29717 km	Semi-major axis
T	14h 10min	Revolution time

Galileo's signal structure

Table 2.6: Galileo frequencies

Frequency [MHz]	Name	Modulation Type	Usage
1575.42	L1	BOC(1,1)	civilian
1176.45	E5a	BPSK(10)	civilian
1207.14	E5b	BPSK(10)	civilian
1191.795	E5	AltBOC(15,10)	civilian
1278.75	E6	BPSK(5)	civilian

Control segment

The control segment is composed of

- 2 Ground Control Center (GCC), Oberpfaffenhofen (Germany), Fucino (Italy)
- 5 satellite control stations (TCC) for communication
- 30 Galileo Sensor Stations (GSS), which receive the signal and process it every ten minutes
- 9 Up Link Stations (ULS), for updating navigation data

2.4.3 Other GNSS

Other GNSS are the Russian GLONASS or the Chinese Beidou-2, which will not be considered in detail below.

The Russian **GLONASS** was developed by the former Soviet Union during the cold war. Today it is operated by the Russian Space Forces, a department of the Russian government. Developed in 1976 it was operational in 1991 with 24 satellites in space. System specifications are referring to [5].

The Chinese **Beidou-2** (see [1]) also known as COMPASS is still under construction today. The system will consist of 35 satellites and will offer services to customers in the Asia-Pacific region in 2012, the global system should be finished in 2020. The interesting thing is that, on the contrary to the previously mentioned systems, 5 of the 35 satellites are in a geostationary orbit, 27 in a Medium Earth Orbit (MEO) and 3 in an Inclined Geostationary Orbit (IGSO). This constellation will provide global coverage.

2.5 GNSS observable and error sources

As mentioned before, GNSS positioning is based on ranging measurements between satellites and receivers. Two kind of measurements can be done, code measurement or pseudorange measurement, both determining a pseudorange between satellite and receiver. The measurement is called pseudorange, because it is not possible to measure the slant range.

The observation equation of code measurement can be described as (see Keller (2007, [10] Chapter 3))

$$\rho_i = c\Delta t_i \tag{2.3}$$

where c is the speed of light, Δt_i is the measured travel time of the signal and ρ_i the pseudorange, for each satellite i.

The signal is corrupted by several error sources, on the one hand, error sources effected by the travel path of the signal and on the other hand errors because of clock bias and orbit determination errors. Substituting these error sources gives the following result

$$\rho_i = R + dR + c(dt - dT) + \iota_i^{group} + \tau + \eta_i \tag{2.4}$$

where

- R slant range between satellite and receiver
- dR orbital error
- c speed of light
- dt receiver clock error
- dT satellite clock error
- ι_i^{group} ionospheric group delay
- τ tropospheric delay
- η noise.

The observation equation for phase measurement (in meters)

$$\Phi_i = R + dR + c(dt - dT) + N_i\lambda_i + \iota_i^{phase} + \tau + \eta_i \tag{2.5}$$

with additional parameters

- N_i integer value of ambiguities
- λ_i wavelength of carrier frequency
- ι_i^{phase} ionospheric phase delay

2.5.1 Ionospheric refraction

The ionosphere is an important error source affecting the range measurements for GNSS (see Klobuchar (1996, [12]) and Keller (2009, [11])). Because of the variability of the Earth's ionosphere it is very difficult to model, so the ionospheric range error can vary from a few meters up to fifty meters. The molecules of the ionosphere get ionized due to ultraviolet radiation of the sun. This effects the ionosphere to be a dispersive medium which means, the refractive index is a function of the frequency of the GNSS signal. Measuring on two or more frequencies, it is possible to determine the first-order ionospheric range error and correct the measured range.

To determine the ionospheric error, one has to distinguish between code measurement, where the signal's propagation velocity is the group velocity and carrier phase measurements, propagating with phase velocity.

The ionospheric error term ι_i for frequency i is

$$\iota_i = \iota_i^{phase} = -\iota_i^{group} = 40.3\mathrm{Hz}^2\mathrm{m}^3\frac{\mathrm{TEC}}{f_i^2} \tag{2.6}$$

With the knowledge of the current Total Electron Content (TEC) of the ionosphere it is possible to reduce the receiver measurement by the ionospheric error.

As said above, it is very difficult to model the ionosphere, but it is possible to determine the TEC by multi frequency measurements and combining them afterwards. These frequency combinations will be discussed in chapter 2.6.

2.5.2 Tropospheric refraction

In contrast to the ionosphere, the troposphere changes it's behavior much slower, it is easier to model and not frequency dependent (see Spilker (1996, [9]) and Keller (2009, [11])). The tropospheric error at zenith is about two or three meters and it consists mainly of two effects. The first and larger effect is the influence of the dry atmosphere caused by N_2 and O_2. It varies with the local temperature and atmospheric pressure in a predictable way. The second effect is the wet atmosphere or water vapor effect which is clearly smaller than the dry one and can't be modeled in such a good way as the dry component.

2.5.3 Other ranging error sources

Other error sources (see Parkinson (1996, [15]) and Keller (2007, [10] Chapter 4)) can be grouped into the following classes.

Table 2.7: Ranging error sources

Group	Parameter	Description
Ephemeris errors	dR	Errors in the transmitted location of the satellite
Satellite clock	dT	Error of the satellite clock
Receiver errors	dt	Errors in the receiver's measurement caused by internal clock bias and software accuracy
Noise and multipath effects	η	Errors because of thermal noise and multipath-effects

2.6 Frequency combinations

If measurements by one and the same receiver are simultaneously done on more than one frequency, it is possible to combine these measurements. This will lead to new frequencies, which have special characteristics.

In the following, the combination of only two frequencies will be shown (see Keller (2007, [10] Chapter 3)).

2.6.1 Ionosphere-free Combination

The ionosphere-free Combination consists of the geometrical components, the orbital errors and the clock errors, but it is free of ionospheric refraction.

$$\begin{aligned}\rho_I &= -\frac{f_1^2}{f_2^2 - f_1^2}\rho_1 + \frac{f_2^2}{f_2^2 - f_1^2}\rho_2 \\ &= R + dR + c(dt - dT) - \frac{f_1^2}{f_2^2 - f_1^2}\eta_1 + \frac{f_2^2}{f_2^2 - f_1^2}\eta_2\end{aligned} \tag{2.7}$$

for code-measurement and for phase-measurement

$$\begin{aligned}\Phi_I &= -\frac{f_1^2}{f_2^2 - f_1^2}\Phi_1 + \frac{f_2^2}{f_2^2 - f_1^2}\Phi_2 \\ &= R + dR + c(dt - dT) - \frac{f_1^2}{f_2^2 - f_1^2}(N_1\lambda_1 + \eta_1) + \frac{f_2^2}{f_2^2 - f_1^2}(N_2\lambda_2 + \eta_2)\end{aligned} \tag{2.8}$$

2.6.2 Geometry-free Combination

The Geometry-free Combination will lead to a new frequency, without the influence of the changing satellite-receiver geometry, orbital errors and clock errors.

$$\rho_G = \rho_1 - \rho_2 = 40.3\mathrm{Hz}^2\mathrm{m}^3 TEC\left(\frac{1}{f_2^2} - \frac{1}{f_1^2}\right) + \eta_1 - \eta_2 \tag{2.9a}$$

$$\Phi_G = \Phi_1 - \Phi_2 = N_1\lambda_1 - N_2\lambda_2 + 40.3\mathrm{Hz}^2\mathrm{m}^3 TEC\left(\frac{1}{f_1^2} - \frac{1}{f_2^2}\right) + \eta_1 - \eta_2 \tag{2.9b}$$

2.6.3 Ionospheric refraction

To calculate the influence in pseudorange of the ionosphere propagation delay two methods can be used.

One way is to calculate the ionospheric refraction via TEC (Total Electron Content)

$$\mathrm{TEC} = \frac{\rho_1 - \rho_2}{40.3\left(\frac{1}{f_1^2} - \frac{1}{f_2^2}\right)} \tag{2.10a}$$

$$\iota_i = \iota_i^{phase} = -\iota_i^{code} = 40.3\mathrm{Hz}^2\mathrm{m}^3\frac{\mathrm{TEC}}{f_i^2} \tag{2.10b}$$

and the other way is to use a frequency combination.

$$\iota_1^{code} = \frac{f_2^2}{f_2^2 - f_1^2}\rho_1 - \frac{f_2^2}{f_2^2 - f_1^2}\rho_2 \tag{2.11a}$$

$$\iota_1^{phase} = -\frac{f_2^2}{f_2^2 - f_1^2}\Phi_1 + \frac{f_2^2}{f_2^2 - f_1^2}\Phi_2 \tag{2.11b}$$

Both deliver the same result and can be used to model the influence of the ionosphere to the signal.

2.7 Stochastic processes

A stochastic process is a mathematical description of a time dependent random processes. To describe it's future evolution probability distributions are used. This means, knowing the initial conditions, doesn't give information where the process will go to, but some paths are more likely than others.

A special case of stochastic processes are stationary processes, which will be used further on.

2.7.1 Stationary processes

Stationary processes are processes whose joint probability distribution does not change when shifted in time or space. This means, the time series has the same expectation value and the same variance over time.

A stochastic process is called stationary in the wide sense (see Meier and Keller (1990, [18])), if

$$E\{X_t\} = m \tag{2.12}$$

the expectation value E is the average value m of the process and if

$$E\{(X_{t_1} - m)(X_{t_2} - m)\} = E\{(X_{t_{1+\tau}} - m)(X_{t_{2+\tau}} - m)\}. \tag{2.13}$$

In this case, the auto covariance function is only dependent on the difference of the arguments

$$E\{(X_{t_1} - m)(X_{t_2} - m)\} = C(t_1 - t_2) \tag{2.14}$$

The connection between auto covariance function and spectral power density for stationary processes is given by the Fourier transformation

$$S(\omega) = \mathcal{F}\{C\} \tag{2.15}$$

For stationary processes the following estimators are available. The empirical covariance function

$$\hat{m} = \frac{1}{N}\sum_{i=1}^{N} X_{t_i} \tag{2.16}$$

$$\hat{C}(n) = \frac{1}{N-n}\sum_{i=1}^{N-n}(X_{t_i} - \hat{m})(X_{t_{i+\tau}} - \hat{m}) \tag{2.17}$$

and an estimation of the empirical power density

$$\hat{S}(\omega) = \mathcal{F}\{X_t\}(\mathcal{F}\{X_t\})^* = |\mathcal{F}\{X_t\}|^2 \tag{2.18}$$

where $(\mathcal{F}\{X_t\})^*$ is the complex conjugate of $\mathcal{F}\{X_t\}$. This estimation is just possible, if the correlation length is small in relationship to the length of the data.

3 Signal conversion

The Septentrio receiver records data in *.sbf[1] format, which can be converted into an ASCII format with the `SBF Converter`. Here the data blocks `MeasEpoch` are interesting, because they contain the measured pseudorange both in code and phase measurement.

3.1 Data recording

The data recording took place at Thales corporation. There are two multi-constellation 2 frequency receivers (PolaRx3_TR) by Septentrio available. Each receiver is able to record two previously defined frequencies, e.g.

1. L1/L2 GPS + Glonass or L1/L2c GPS
2. L1/L5 GPS + E1/E5a Galileo (GIOVE)

The data can be recorded with up to 10Hz.

The aim of the study work is the analysis of the modernized GPS (L5) and Galileo (E1, E5a) signal. This has to be considered in the data recording, because the modernized GPS signal is currently only broadcasted by two satellites (PRN 01 and PRN 25) and at this time, the satellite PRN 01 has the status unhealthy. Thus L5 signals can be distorted or not received.

The Galileo signal can be received by the two test satellites (GIOVE-A and GIOVE-B). To set the recording time, the website `http://www.galtecproject.de` provided by Thales can be used. It gives an overview of the available satellites to different times. It is important to ensure that the recording period is timed to see the Galileo satellites as long as possible.

3.1.1 Recorded data

The following data sets have been acquired and will be used for further investigations.

Table 3.1: Recorded Data

Origin	Recording date	Receiver	Filename	Frequencies
Thales	19.07.2010	LMU	LMU_STGT2000	L1/L5, E1/E5
	25.07.2010	LMU	LMU_STGT2060	L1/L5, E1/E5
	22.08.2010	AAU	AAU_SEPT2340	L1/L2

3.2 Conversion to an ASCII format

The conversion of the *.sbf files to an ASCII format can be done by using the tool `SBF converter`. To do this, run the tool, choose the file to be converted and in the 'convert to' field select 'STF'.

[1] sbf: Septentrio Binary Format

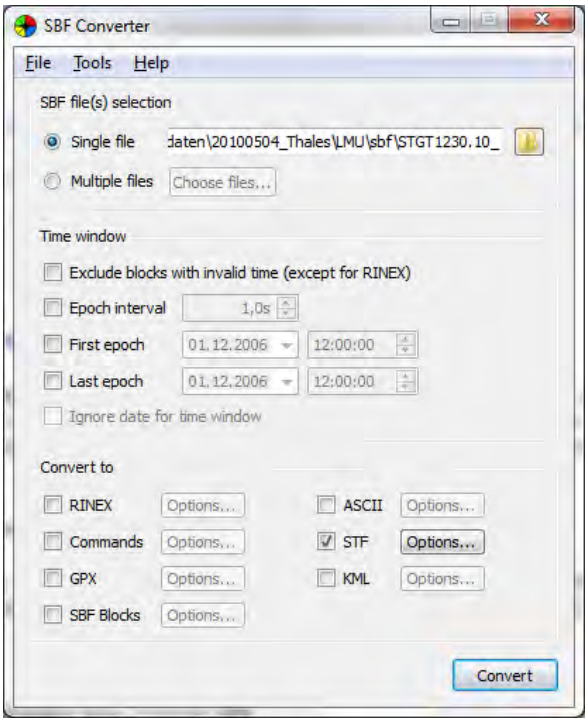


Figure 3.1: SBF Converter

In addition to the field 'Time window' the data can be pre filtered, e.g. for the output interval.

After that one has to define, which records are to be issued. This can be defined under 'Options'. Particularly 'MeasEpoch', 'PVTGeodetic v2' and 'GPSAlm' are required.

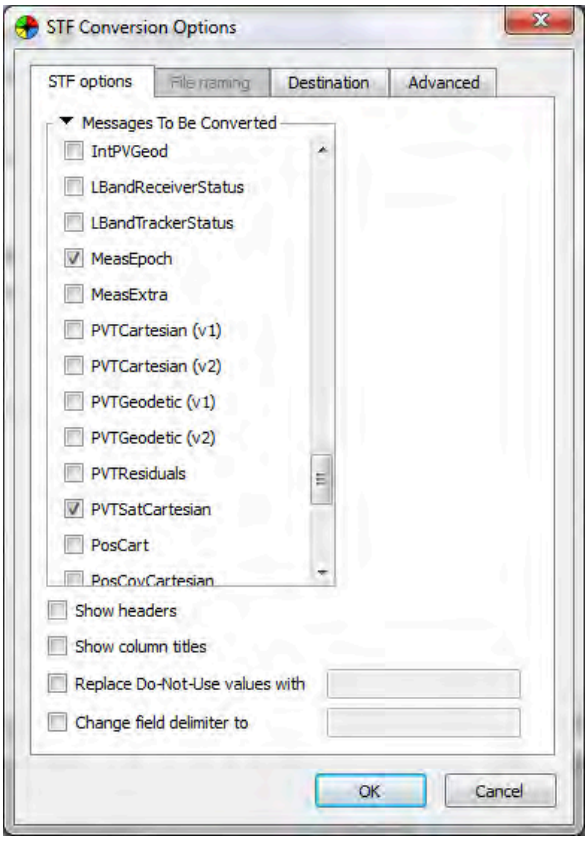


Figure 3.2: SBF Settings

With the option 'Show column titles' one can write the column definitions into the output file. To load the data into Matlab, the column definitions must be hidden. There must be only the pure measurement data in the file.

After all options are set, the conversion can be started.

3.2.1 Column definition

Here is an overview of the column definitions of the files, which will be used for our investigations. These definitions will be also taken for the *.mat[2] files, after the import.

MeasEpoch

1. WNc [week]
2. TOW [s]
3. RxChannel
4. SVID
5. FreqNr
6. Antenna
7. SignalType
8. Code [m]
9. Carrier [cycle]
10. Doppler [Hz]
11. C/N0 [dB-Hz]
12. LockTime [s]
13. HalfCycleAmbiguity
14. Smoothing

[2] mat: Matlab specific data format

PVTGeodetic

1. WNc [week]
2. TOW [s]
3. Mode
4. 2D/3D
5. Error
6. NrSV
7. Latitude [rad]
8. Longitude [rad]
9. Height[m]
10. Vn [m/s]
11. Ve [m/s]
12. Vu [m/s]
13. ClockBias [ms]
14. ClockDrift [ppm]
15. ReferenceID
16. MeanCorrAge [s]
17. NrBases
18. Undulation [m]
19. COG [deg]
20. Datum
21. TimeSystem
22. SignalInfo
23. WACorrInfo
24. AlertFlag
25. AutoBase

GPSAlm

1. WNc [week]
2. TOW [s]
3. PRN
4. E
5. t_oa
6. Delta_i [semi-circle]
7. OMEGADOT [semi-circle/s]
8. SQRT_A [mû.5]
9. OMEGA_0 [semi-circle]
10. omega [semi-circle]
11. M0 [semi-circle]
12. a_f1 [s/s]
13. a_f0 [s]
14. WN_a [week]
15. AS
16. health8

for additional information see [16].

3.3 Matlab data import

Once the data is converted to an ASCII format, it can now be imported into Matlab and be stored in a *.mat file for further use. This file can be read into Matlab very quickly to avoid unnecessary delays in working with large amounts of data. A corresponding Matlab function might look like this

```
%%% read data in mat file
% after loading mat file, content is stored in variable data
% inFile: Path of input data as string
% outFile: Path of output data as string
% Example:
% inFile = 'data/STGT1230.10_PVTGeodetic_2_5secInterval.stf';
% outFile = 'data/STGT1230.10_PVTGeodetic_2_5secInterval.mat';
% DataName = 'PVTGeodetic'

function readData(inFile, outFile, DataName)

%%% read data and save as *.mat file
var = genvarname(DataName);
data = csvread(inFile);
eval([var ' = data;']);
save(outFile, DataName);
```

and to read the *.mat file

```
%%% load measEpoch
load data/STGT1230.10_MeasEpoch_5secInterval.mat
measEpoch = data;
clear data
```

Now, the measured data is cached in the matrix 'measEpoch'. Columns are defined as in Section 3.2.1.

3.4 Staircase structure in code and phase measurement

Visualizing the code and phase measurement of imported data, we get a staircase structure, which is not natural and might occur because of receiver errors, which have to be removed.

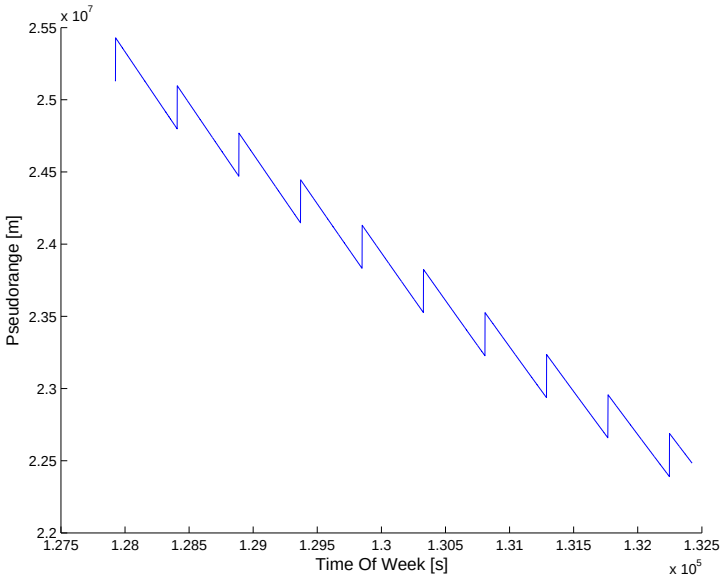


Figure 3.3: Extract of the code measurement with staircase structure

3.4.1 Analysis

This staircase structure occurs because of the receiver clock, which is not as stable, as an atomic clock. Thus we get a receiver clock drift. To compute the current offset of the receiver time and the GNSS time we have to subtract them from each other.

$$\Delta t = t_{GNSS} - t_{Receiver} \tag{3.1}$$

With t_{GNSS} GNSS system time and $t_{Receiver}$ the receiver cock time, which is drifting heavily.

By default, a threshold of $\Delta t = 0.5$ms is set for the receiver. If this threshold is reached, the receiver time is reduced by 1ms. These receiver clock corrections are not considered in the recorded pseudorange, the result is shown in the figure above.

To remove this staircase pattern from the measured data, two approaches were analyzed.

3.4.2 Tested reduction methods

Equating the measured values

Here, the idea was to detect the positions of which jumps occur and, based on this knowledge, to reduce the following data by setting them to the last value, before the jump.

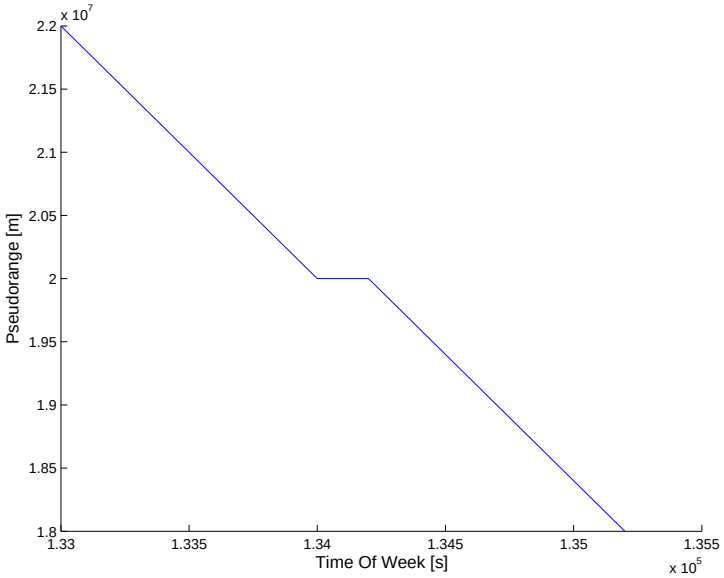


Figure 3.4: Elimination of step structure through equating

The problem with this method can be seen clearly in the above figure. The staircase structure was indeed significantly reduced, however, discontinuities still appear in the corrected data. This result is not satisfactory.

Reduction by receiver clock correction

This method is based on the clock correction described in section 3.4, which automatically takes place in the receiver and which is logged.
Logging of the clock bias can be found in the 'PVTGeodetic v2' measurement block. Here one can find the computed clock bias of the receiver clock to each epoch. With this knowledge, it is possible to calculate the range error, effected by the clock bias

$$\Delta\rho = -c\Delta t[\mathrm{m}] \tag{3.2}$$

for the code measurement, where Δt is the clock bias.

To convert the range error into cycles, which is the unit of the phase measurement, the following context is used

$$\Delta\Phi_i = -\frac{\Delta\rho}{\lambda_i} \tag{3.3}$$

$$\lambda = \frac{c}{f_i} \tag{3.4}$$

with $c = 2.99792458 \cdot 10^8 m/s$ speed of light (according to [13]) and the relevant frequency of the carrier signal f_i.

After applying the calculated ranging errors to the original measurement, we get a corrected pseudorange measurement in code and carrier observation.

$$\rho_{corr} = \rho + \Delta\rho \tag{3.5a}$$

$$\Phi_{corr} = \Phi + \Delta\Phi_i \tag{3.5b}$$

The big advantage of this method is the knowledge of the actual clock bias which is logged. With this information it is possible to correct the data with the error it was distorted. Thus this method is independent from previous measurements like the method described above and seems to be the best method to correct the measurements.

After applying, one gets the following result

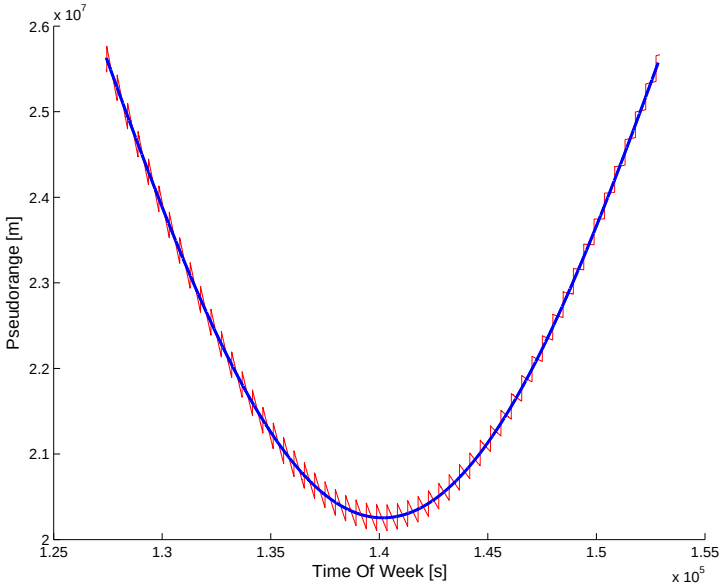


Figure 3.5: Removal of staircase structure

Red, the original signal, and blue, the signal after the removal of the clock bias.

The developed algorithm automatically corrects all measurements from code and phase observation for all satellites contained in 'MeasEpoch'.

An example of corrected code and phase observation

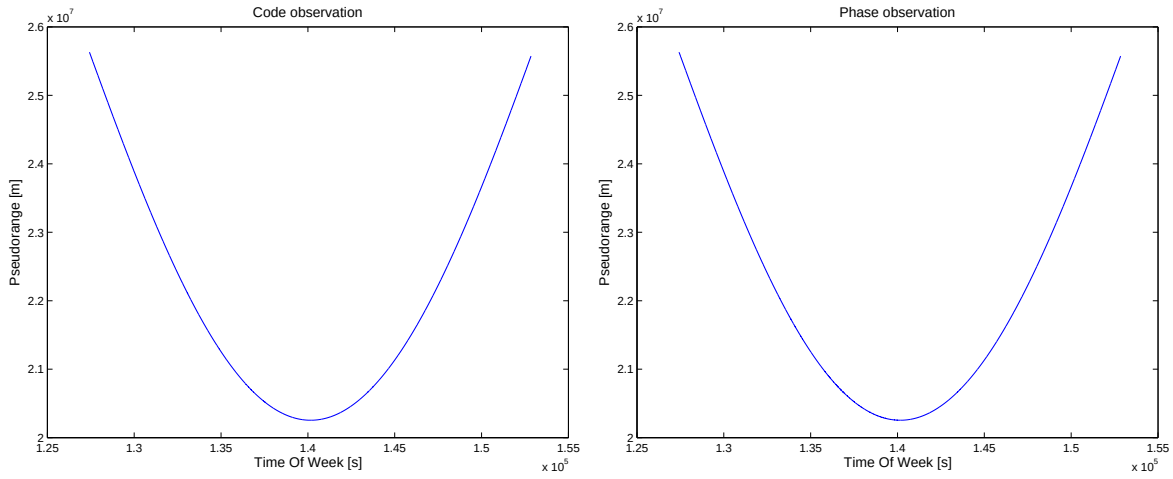


Figure 3.6: Corrected code and phase observation for GPS PRN 25

For further applications, the observation equation, defined in chapter 2.5, for pseudorange measurement can be re-written without the receiver clock error dt, if the algorithm described above was applied.

$$\rho_i = R + dR + c \cdot dT + \iota_i^{group} + \tau + \eta_i \tag{3.6a}$$

$$\Phi_i = R + dR + c \cdot dT + N_i\lambda_i + \iota_i^{phase} + \tau + \eta_i \tag{3.6b}$$

3.5 Synchronization of epoch

The data recorded by the receiver is now reduced of the errors caused by the receiver clock bias. However, the data can not be compared directly, since it may not always be consistent at the same epochs. It is possible that for one and the same satellite at two different recorded frequencies, different data, to different epochs is available.

An example will illustrate this:

Satellite 1

ϕ_{f_1} - Epoch 1-10

ϕ_{f_2} - Epoch 1-5 and 8-15

This is likely because the signal is dropping at these times. The continuation of the recording is done later, if the signal is back.

Therefore, it is required to intersect the measured data to only get data at epochs, where it is available on both frequencies. This functionality was implemented in the Matlab function `syncEpoch` (see A.1.1) and can be called as follows

```
%%% syncEpoch
[measEpoch] = syncEpoch(measEpoch);
save('data/STGT1230.10_MeasEpoch_5secInterval_syncEpoch.mat','measEpoch')
```

The function `syncEpoch` automatically matches all data for all satellites and frequencies in the data file.

3.6 Plausibility check

The original data must be optimized by the above algorithms to prepare it for evaluation, at this point a plausibility check of the reduced data and the used algorithms is essential. These checks ensure that the reduced data is suitable for further use.

3.6.1 Visual inspection

Looking at the corrected data at the positions of the discontinuities, one cannot find any irregularities in the data. From this perspective, the data seems to be sufficiently corrected.

3.6.2 Comparison of reduced and original data

The observation equations for code and phase measurement are

$$\rho_i = R + dR + c \cdot dT + \iota_i^{group} + \tau + \eta_i \tag{3.7a}$$

$$\Phi_i = R + dR + c \cdot dT + N_i\lambda_i + \iota_i^{phase} + \tau + \eta_i \tag{3.7b}$$

for all observations, which are modified by the algorithms. For the original measurement, we have to consider the receiver clock error dt again.

$$\rho_i' = R + dR + c(dt - dT) + \iota_i^{group} + \tau + \eta_i \tag{3.8a}$$

$$\Phi_i' = R + dR + c(dt - dT) + N_i\lambda_i + \iota_i^{phase} + \tau + \eta_i \tag{3.8b}$$

While taking differences either of code measurement ρ_i or phase measurement Φ_i at different frequencies, the range R, the satellite clock error dT and receiver clock error dt, in case of the original measurement, will disappear.

If all used algorithms work correct, the differences between original data and reduced data must be the same, or if we subtract the differences from each other, they must be zero.

$$(\rho_1 - \rho_2) - (\rho_1' - \rho_2') \stackrel{!}{=} 0 \tag{3.9a}$$

$$(\Phi_1 - \Phi_2) - (\Phi_1' - \Phi_2') \stackrel{!}{=} 0 \tag{3.9b}$$

This was verified using the code measurement for a satellite.

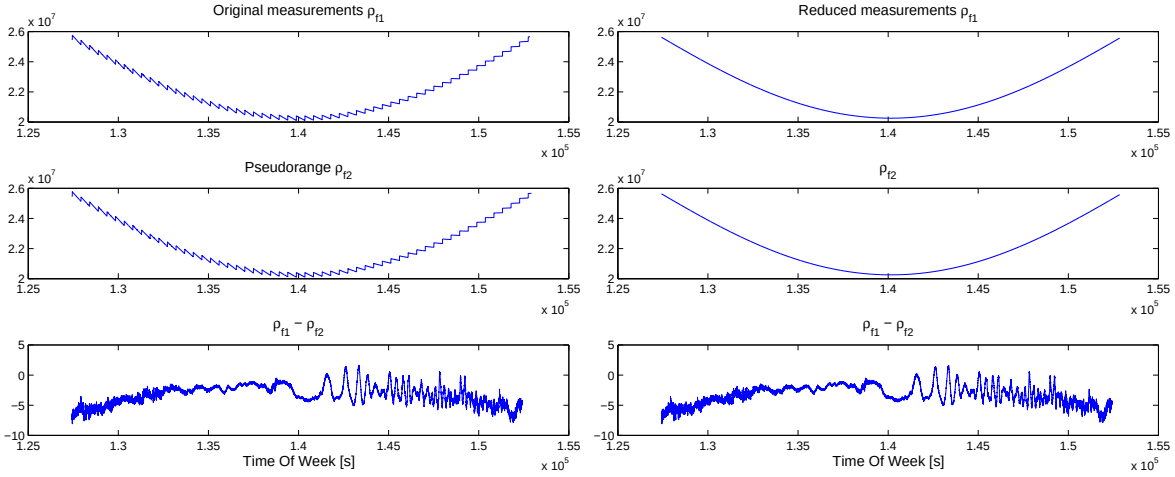


Figure 3.7: Comparison of the reduced data with the original

If we now subtract both differences from one another, they must cancel each other out, providing no further errors remaining in the data.

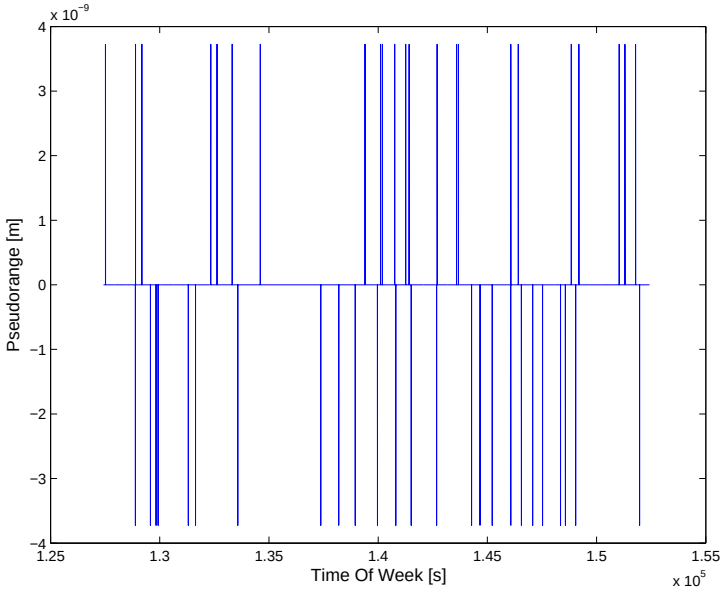


Figure 3.8: Difference of both methods

Looking at the result, we can find variations in the difference, but they are very small and can be interpreted as numerical noise.

3.6.3 Result

The result of this plausibility check shows, that the measured data is processed correctly in the applied algorithms and thus, they are internally consistent. This is not an absolute control of the data, the receiver records, but a verification of the algorithms from chapter 3.4 and 3.5.

3.7 Computation of elevation

To compute the elevation angle between receiver and satellite, it is necessary to know the position of the satellite in the Earth-fixed coordinate system, in which the receiver position is known.

These coordinates will be calculated using Kepler elements. To get a more precise knowledge of the satellite's position it is possible to use osculating Kepler elements, which consider disturbance forces, but for our investigations, it is sufficient to use just Kepler elements.

The receiver used does also log almanac information. So this will be the source to get the Kepler elements of the satellites. Unfortunately the receiver is just logging almanac information for GPS satellites, nor for Galileo satellites. The substantiation of the manufacturer is, that at the moment, Galileo satellites do not transmit the complete almanac message, that is why the receiver is not able to log this information.

To get around this problem, we decided to use TLE[3] data, provided by `http://www.celestrak.com`. The TLE format also provides Kepler elements, but in another data format. The idea was to transfer these TLE data to the almanac format, for use of the same algorithms processing the Earth-fixed coordinates. Special attention has to be given to the transformation of the time component, which will be described later.

[3] TLE: Two Line Element, A data set, which provides orbital parameters

3.7.1 GPS - Almanac format

A short description of the almanac format recorded by the Septentrio receiver (see [16]), the included parameters and their unit.

Table 3.2: Almanac format definition

Parameter	Unit	Description
WNc	[week]	GPS week number
PRN		PRN number of satellite of which the almanac is given
e		Mean eccentricity
t_{oa}	[s]	Almanac reference time of week
ΔI	[semi-circles]	Inclination angle at reference time, relative to $I_0 = 0.3[\text{semi} - \text{circles}]$
$\dot{\Omega}$	[semi-circles/s]	Rate of right ascension
$\sqrt{A}$	$[\text{m}]^{1/2}$	Square-root of the semi-major axis
Ω_0	[semi-circles]	Longitude of ascending node
ω	[semi-circles]	Argument of perigee
M_0	[semi-circles]	Mean anomaly
a_{f1}	[s/s]	SV clock drift
a_{f0}	[s]	SV clock bias
WN_a	[week]	Almanac reference week, to which t_{oa} is referenced

Because the receiver used to record the data does not log the rate of right ascension $\dot{\Omega}$ it has to be calculated additionally

$$\dot{\Omega}(t_k) = \frac{3}{2} n_0 C_{20} \left(\frac{R_E}{a}\right)^2 \frac{\cos I_0}{(1-e_0^2)^2} t_k \tag{3.10}$$

with n_0 the mean angular velocity and C_{20} the Earth flattening.

3.7.2 TLE - Format description

The TLE format consists in reality of three lines, in the first line we find the name of the satellite, the following two data-lines contain the orbital elements (see [6]).

An example of GIOVE-A and GIOVE-B:

```
1 GIOVE-A
2 1 28922U 05051A   10172.86766662  .00000053  00000-0  10000-3 0  6980
3 2 28922  56.1040 147.6654 0010231 342.1076  17.8225  1.69469557 27831
4 GIOVE-B
5 1 32781U 08020A   10174.99557339  .00000075  00000-0  10000-3 0  3220
6 2 32781  55.9247 182.2917 0021691 225.2350 134.6185  1.70949862 13483
```

Specification of the format

Row 1 contains only the satellite name.

Row 2:

Table 3.3: TLE format definition, Row 2

Value	Parameter	Unit	Description
1			Row number
28922			NORAD[4] Catalog number
U			Classification (U = unclassified)
05051A			International description, 05 last two digits of launch year, 051 launch number of the year, 'A' piece of the launch
10172.86766662			10 Epoch year (last two digets of year), 172.86766662 Epoch (Day of year and fractional portion of day)
.00000053	$\dot{n}$	$[d^{-2}]$	First time derivative of mean motion
00000-0	$\ddot{n}$	$[d^{-3}]$	Second derivative of mean motion
10000-3	B^*	[Earth-radii]	BSTAR drag term
0			Originally this should have been 'Ephemeris type'
698			Element number
0			Checksum

Row 3:

Table 3.4: TLE format definition, Row 3

Value	Parameter	Unit	Description
2			Row number
28922			NORAD Catalog number
56.1040	I	[degrees]	Inclination
147.6654	Ω_0	[degrees]	Right ascension of ascending node
0010231	e		Eccentricity (decimal point assumed)
342.1076	ω	[degrees]	Argument of perigee
17.8225	M_0	[degrees]	Mean anomaly
1.69469557	n	[revs per day]	Mean motion
2783		[revs]	Revolution number at epoch
1			Checksum

To transfer the TLE parameters to almanac format, it is necessary to compute the semi-major axis

$$a = \left(\frac{\mu}{n^2}\right)^{1/3} \tag{3.11}$$

and again, the rate of right ascension $\dot{\Omega}$ (see equation 3.8) as described above.

Other parameters are the same as in the almanac format, just in another unit, so one has to adapt that.

But an other important task is to transfer the given time information (Epoch year and day) into almanac reference week WN_c and reference time of week t_{oa}, done in respective algorithms (see A.2).

3.7.3 From almanac to xyz

Knowing the almanac parameters for each satellite, it is possible to compute the Earth-fixed coordinates of the satellites (see [13] Pages 96-98).

$\mu = 3.985005 \cdot 10^{14} \mathrm{m}^3/\mathrm{s}^2$	WGS84 value of the Earth's universal gravitational parameter for GPS user
$\dot{\Omega}_e = 7.2921151467 \cdot 10^{-5} \mathrm{rad/s}$	WGS84 value of the Earth's rotation rate
$A = \left(\sqrt{A}\right)^2$	Semi-major axis
$n = \sqrt{\frac{\mu}{A^3}}$	Computed mean motion
$t_k = t - t_{oe}^*$	Time from ephemeris reference epoch
$M_k = M_0 + nt_k$	Mean anomaly
$E_k = e \sin E_{k-1} + M$	Eccentric anomaly
$\nu_k = \tan^{-1}\left(\frac{\sqrt{1-e^2}\sin E_k/(1-e\cos E_k)}{(\cos E_k - e)/(1-e\cos E_k)}\right)$	True anomaly
$u_k = \nu_k + \omega$	Argument of latitude
$r_k = A(1 - e\cos Ek)$	Radius
$I_k = I_0 + \Delta I_k$	Inclination
$\Omega_k = \Omega_0 + \left(\dot{\Omega} - \dot{\Omega}_e\right) t_k - \dot{\Omega}_e t_{oa}$	Longitude of ascending node

* t is GPS time at time of transmission. t_k shall be the total time difference between the time t and the epoch time t_{oa} and must account for beginning or end of week crossovers. That is, if t_k is greater than $302,400$s, subtract $604,800$s from t_k. If t_k is less than $-302,400$s, add $604,800$s.

To compute eccentric anomaly (reverse Kepler equation) an iteration is required $M \longrightarrow E$

$$\begin{aligned} E_{i+1} &= e \sin E_i + M \\ E_0 &= 0 \\ E_1 &= M \\ E_2 &= e \sin E_1 + M \\ E_3 &= \dots \end{aligned}$$

The positions in the orbital plane, relative to the perifocal coordinate system

$$x'_k = r_k \cos u_k \tag{3.12a}$$
$$y'_k = r_k \sin u_k \tag{3.12b}$$

and the final Earth-fixed satellite coordinates

$$x_k^s = x'_k \cos \Omega_k - y'_k \cos i_k \sin \Omega_k \tag{3.13a}$$
$$y_k^s = x'_k \sin \Omega_k - y'_k \cos i_k \cos \Omega_k \tag{3.13b}$$
$$z_k^s = y'_k \sin i_k \tag{3.13c}$$

for each time k.

Figure 3.9: Satellite positions in Earth-fixed coordinate system

3.7.4 Elevation

With the knowledge of the satellite's position and the receiver antenna position, both in the same coordinate system (in this case: Earth-fixed), one can compute the elevation by vector multiplication.

$$\cos Z_k = \frac{x_k^S \cdot x^A}{|x_k^S| \cdot |x^A|} \tag{3.14}$$

$$E_k = 90° - Z_k \tag{3.15}$$

with $x_k^S = \begin{bmatrix} x_k^s & y_k^s & z_k^s \end{bmatrix}^T$ the satellite's position at epoch k and $x^A = \begin{bmatrix} x^A & y^A & z^A \end{bmatrix}^T$ the position of the receiver antenna. Subtracting 90° from the zenith distance Z_k leads to the elevation E_k.

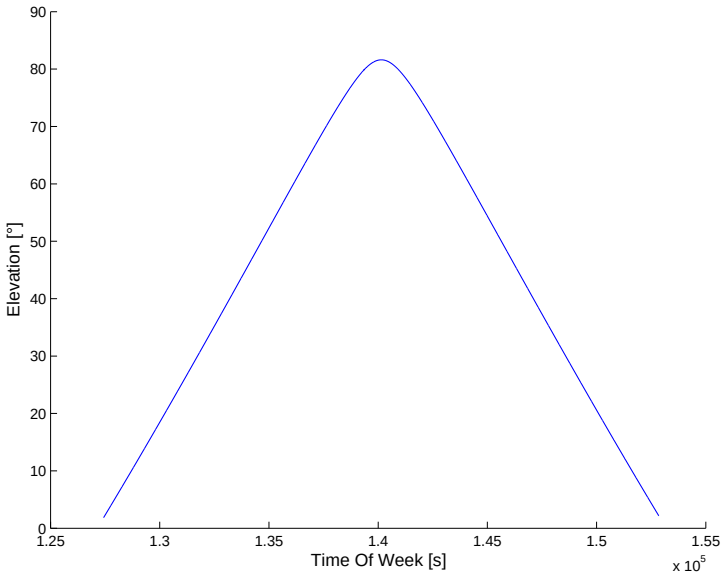


Figure 3.10: Calculated elevation over time

3.8 Final file definitions

After all processing steps are done, a final column definition, with original and additional columns, of the file measEpoch_processed.mat will be done here.

Table 3.5: Final column definition

Column	Abbreviation	Unit	Status	Description
1.	WNc	[week]		The GPS week number associated with the TOW.
2.	TOW	[s]		Time-Of-Week. Time tag, from the beginning of the current Galileo/GPS week.
3.	RxChannel			Receiver channel, on which this satellite is currently tracked (see SBF reference 2.11)
4.	SVID			Satellite ID (see SBF reference 2.9)
5.	FreqNr			For GLONASS satellites, this is the frequency number (see SBF reference 2.9)
6.	Antenna			Antenna ID
7.	SignalType			Signal type (see SBF reference 2.10)
8.	Code	[m]		Pseudorange of code measurement.
9.	Carrier	[m]	updated	Pseudorange of carrier phase measurement.
10.	Doppler	[Hz]		Carrier Doppler (positive for approaching satellites).
11.	C/N0	[dB-Hz]		Carrier to Noise ratio.
12.	LockTime	[s]		Duration of continuous carrier phase.
13.	HalfCycleAmbiguity			Is set, if carrier has a half-cycle ambiguity.
14.	Smoothing			If set, the smoothing filter has reached the requested smoothing interval.
15.	x_E	[m]	new	Satellite coordinate in Earth-fixed system.
16.	y_E	[m]	new	Satellite coordinate in Earth-fixed system.
17.	z_E	[m]	new	Satellite coordinate in Earth-fixed system.
18.	Elevation	[rad]	new	Calculated satellite elevation from antenna position.
19.	ρ_{IR}	[m]	new	Ionospheric refraction (code).
20.	Φ_{IR}	[m]	new	Ionospheric refraction (phase).

4 Extraction of signal noise

To extract the signal noise out of the measurement, the geometrical part of the signal has to be removed. If we look at the observation equation of GNSS after the receiver clock error was removed by the algorithms described in chapter 3.

$$\rho_i = R + dR + c \cdot dT + \iota_i^{group} + \tau + \eta_i \tag{4.1a}$$
$$\Phi_i = R + dR + c \cdot dT + N_i\lambda_i + \iota_i^{phase} + \tau + \eta_i \tag{4.1b}$$

for code and for phase measurements, we find two components, representing the geometrical influence, R, the slant range between satellite and receiver and dR the orbital error of the satellite in direction to the receiver.

To extract these components, two fundamental strategies will be analyzed.

4.1 Data-driven strategy

The data driven strategy is a strategy without assumptions about the behavior of known error sources like ionospheric or tropospheric errors. To reduce the signal by geometry, a polynomial of n-th order should be fitted to the data. After that, the measurement can be reduced by the polynomial and the resulting signal consists only of noise. This strategy is feasible, because all effects, except noise, change slowly in time and noise is of a high frequency. This is the reason, why we tried to fit a polynomial of low order into the data.

4.1.1 Polynomial fit

A polynomial of n-th order can be described in a one dimensional way by the following equation.

$$f(x) = a_0 + a_1x + a_2x^2 + \ldots + a_nx^n \tag{4.2}$$

The adjustment with the method of least squares delivers an estimated polynomial

$$f_{\rho_i}(\hat{x}) \approx R + dR + c \cdot dT + \iota_i^{group} + \tau \tag{4.3a}$$
$$f_{\Phi_i}(\hat{x}) \approx R + dR + c \cdot dT + N_i\lambda_i + \iota_i^{phase} + \tau \tag{4.3b}$$

which will be subtracted from the measurement, to remove the geometrical and longwave components.

$$\rho_i' = \rho_i - f_{\rho_i}(\hat{x}) = \eta_i \tag{4.4a}$$
$$\Phi_i' = \Phi_i - f_{\Phi_i}(\hat{x}) = \eta_i \tag{4.4b}$$

Fitting a polynomial into the signal, has shown that a single polynomial for the whole observation time doesn't estimate a pseudorange measurement in a satisfying way. This is, why the signal was split into pieces of approx. 20 min length and independent polynomials were estimated for these pieces.

The initial pseudorange observation

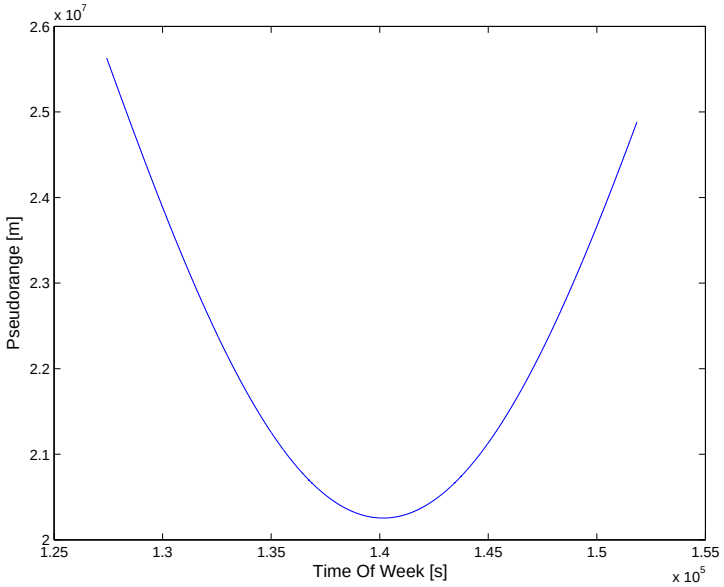
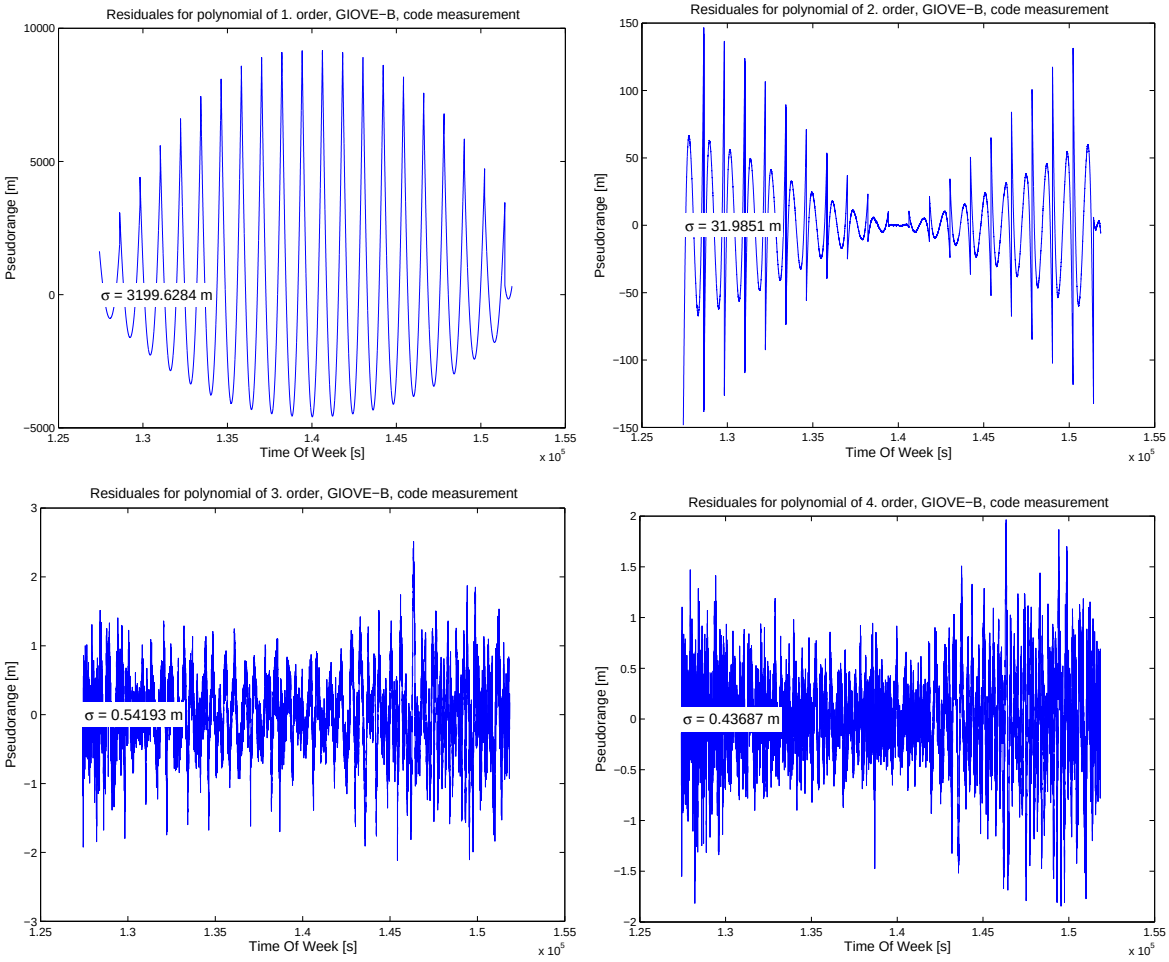


Figure 4.1: Initial pseudorange observation

and the results, shown for the polynomials of different degrees



Looking at the results above, we can see that the polynomials have discontinuities at the positions, the signal was separated. The results of the polynomials with high order were better, but if the order increases, the polynomial begins to oscillate with high frequencies. In contrast to the low frequency, in which the satellite - receiver geometry changes. This is the reason, why polynomials of high order are not suitable to estimate pseudorange measurements.

To avoid the discontinuities of the low order polynomials, it is possible to introduce splines with c_0, c_1 and c_2 continuity at the separation positions. This could be a strategy for further investigations, but will not be discussed within this thesis.

4.2 Model-based extraction

To extract the signal noise using a model, can be done by calculating the ionospheric refraction, which consists of the propagation delay triggered by the dispersive ionosphere and noise.

To get the ionospheric refraction, two strategies can be used, one of them is to compute the current TEC. With the knowledge of the content between ionospheric refraction ι_i and TEC

$$\iota_i = \iota_i^{phase} = -\iota_i^{code} = 40.3\mathrm{Hz}^2\mathrm{m}^3\frac{\mathrm{TEC}}{f_i^2} \tag{4.5}$$

the observation equation can be formed to

$$\begin{aligned}\rho_1 - \rho_2 &= \iota_1 - \iota_2 + \eta_1 - \eta_2 \\ &= 40.3\frac{\mathrm{TEC}}{f_1^2} - 40.3\frac{\mathrm{TEC}}{f_2^2} + \eta_1 - \eta_2 \\ &= 40.3\mathrm{TEC}\left(\frac{1}{f_1^2} - \frac{1}{f_2^2}\right) + \eta_1 - \eta_2\end{aligned}$$

and finally

$$\mathrm{TEC} = \frac{\rho_1 - \rho_2 - \eta_1 + \eta_2}{40.3\left(\frac{1}{f_1^2} - \frac{1}{f_2^2}\right)} \tag{4.6}$$

for code observations. For carrier observations we get

$$\mathrm{TEC} = \frac{\Phi_1 - \Phi_2 - N_1\lambda_1 + N_2\lambda_2 - \eta_1 + \eta_2}{40.3\left(\frac{1}{f_1^2} - \frac{1}{f_2^2}\right)}. \tag{4.7}$$

The other way is to use a frequency combination, which delivers the ionospheric refraction directly

$$\iota_1^{code} = \frac{f_2^2}{f_2^2 - f_1^2}\rho_1 - \frac{f_2^2}{f_2^2 - f_1^2}\rho_2 \tag{4.8a}$$

$$\iota_1^{phase} = -\frac{f_2^2}{f_2^2 - f_1^2}\Phi_1 + \frac{f_2^2}{f_2^2 - f_1^2}\Phi_2 \tag{4.8b}$$

A closer look at the resulting ionospheric refraction shows, that ι is a function of TEC and a constant frequency f. Though TEC is a function of the elevation $E(t)$ and the elevation is a function of the satellite - receiver geometry, both changing in time.

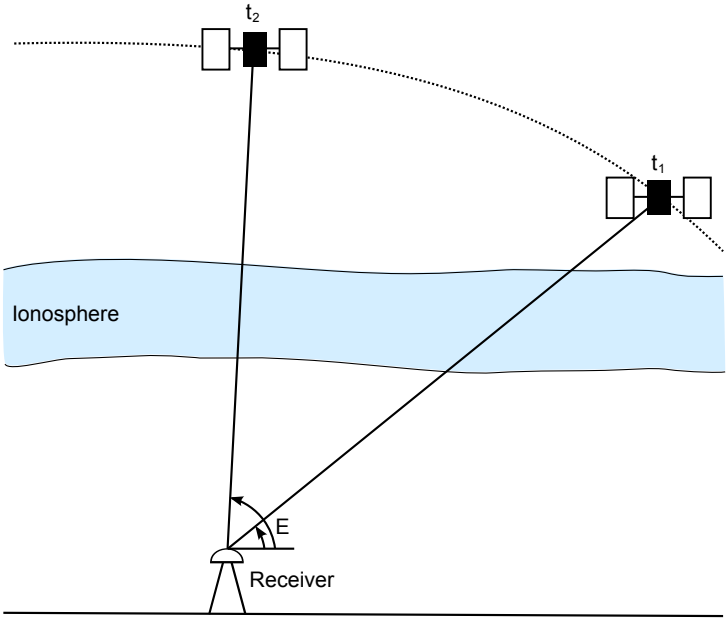


Figure 4.2: Influence of ionosphere to the signal travel path at different elevation angles

What we do get has a structure like

$$\iota(E(t)) + \eta \tag{4.9}$$

with $\iota(E(t))$ the ionospheric refraction, changing slowly in time, plus high frequency noise η.

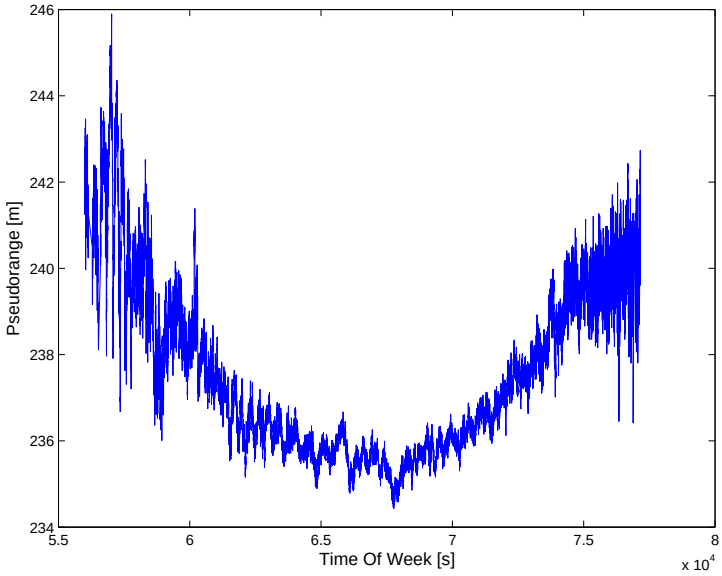


Figure 4.3: Ionospheric refraction

To eliminate the longwave ionospheric error $\iota(E(t))$ and therefore, the remaining geometrical component, a polynomial of low order can be estimated and afterwards subtracted.

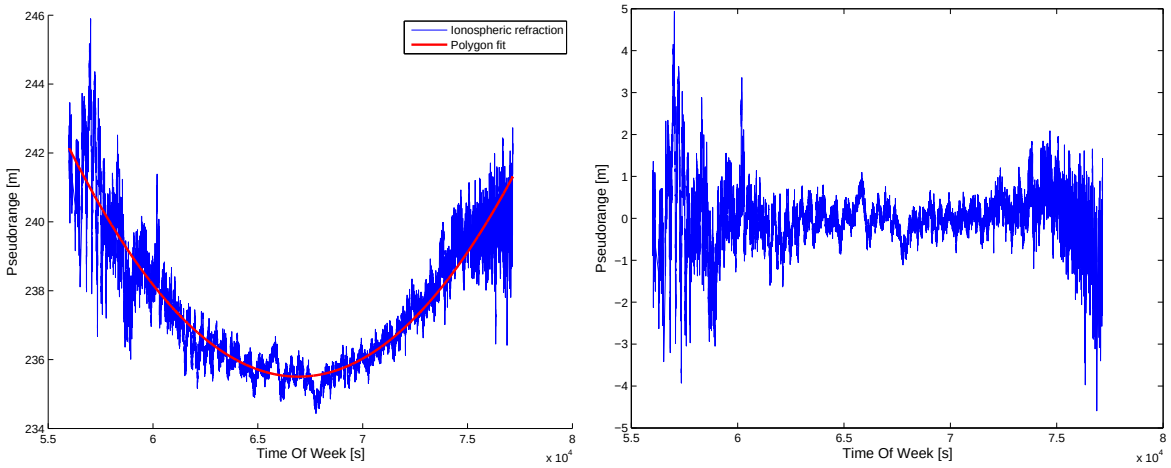


Figure 4.4: Elimination of the remaining geometrical components

Afterwards, we do get a stochastic process η, without geometrical influence. This process can now be analyzed with statistical methods, to get more information about the signal behavior over time.

5 Analysis

To analyze the noise in the signal, the isolated noise data out of chapter 4 will be used.

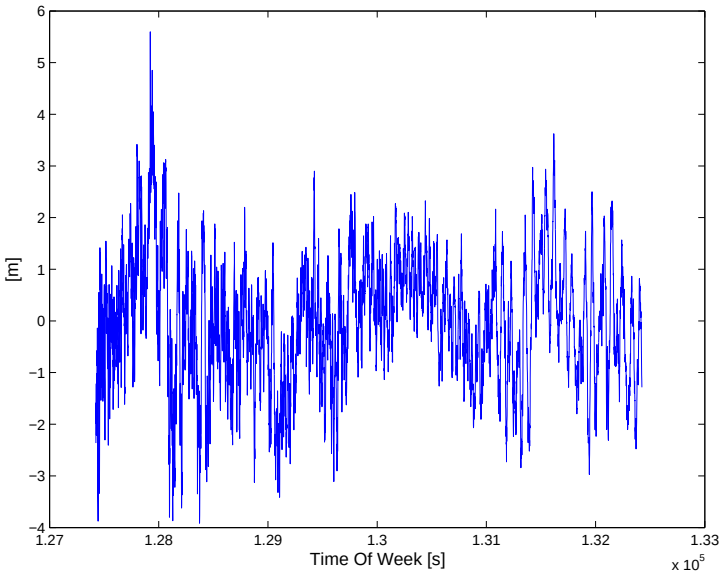


Figure 5.1: Ionospheric refraction without influence of geometry

This process η is not affected by geometry, but isn't stationary yet. To get a stationary process the idea was to split the whole process into n pieces of the same length and compute the standard deviation σ_i for each piece. After a normalization of the standard deviation, each region of the process will be weighted with the inverse of the normalized standard deviation.

$$X_{t,i} = \frac{\eta_i}{\sigma_i} \tag{5.1}$$

where η_i is the region of the process and $X_{t,i}$ the resulting stationary process for $i = 1, 2, ..., n$ parts.

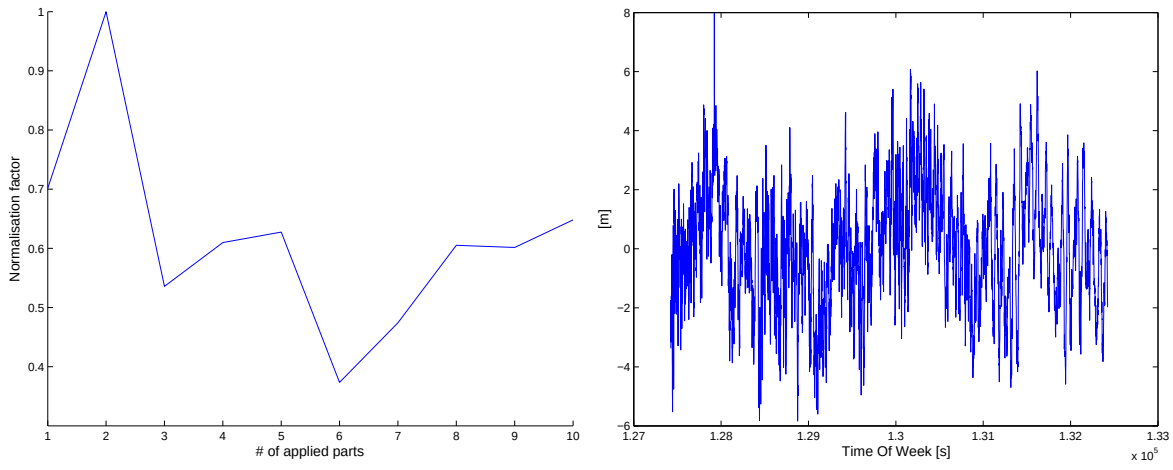


Figure 5.2: Normalization factor and resulting stationary process

5.1 Empirical covariance function

Now it is possible to use the following estimator to compute the covariance function $\hat{C}(n)$

$$\hat{m} = \frac{1}{N} \sum_{i=1}^{N} X_{t_i} \tag{5.2}$$

$$\hat{C}(n) = \frac{1}{N-n} \sum_{i=1}^{N-n} (X_{t_i} - \hat{m})(X_{t_{i+n}} - \hat{m}) \tag{5.3}$$

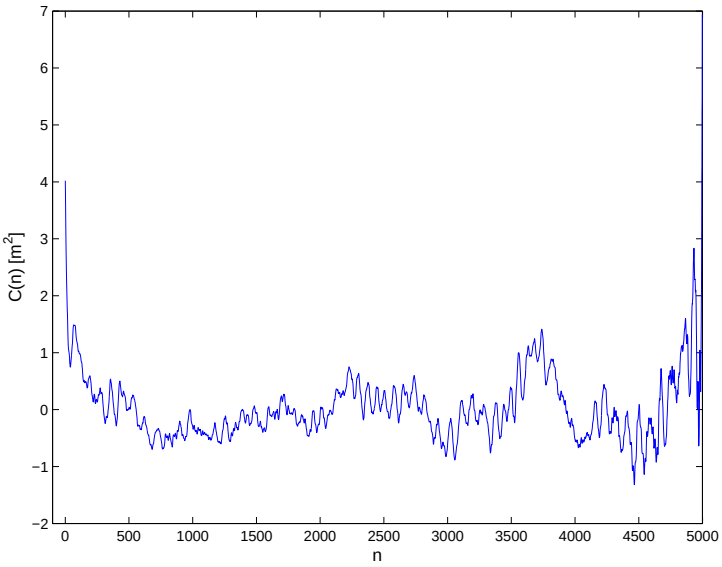


Figure 5.3: Empirical covariance function

At this point, the behavior of the empirical covariance function at the beginning is very interesting, because it is characterizing the noise ratio of the received signal. To be able to compare different signals with each other, an exponential function will be fitted into the data. The resulting estimated parameter of this function will allow a comparison between the received signals.

5.1.1 Approximating the covariance function

The fit is based on the assumption, that the normalized noise is the sum of white noise with the variance a_0 and a Markov process with the variance a_1 and a parameter a_2, describing the decay behavior and thus the correlation length. So the following model will be used

$$y = f(x) = a_0 + a_1 e^{a_2 x} \tag{5.4}$$

Based on the assumption that the estimated process is a sum of white noise and a Markov process, the parameter a_0 will only be determined for $C(0)$, because the covariance function of white noise is a delta function with a value at $C(0)$ and $C(n) = 0$ for $n \neq 0$.

$$a_0 \begin{cases} = 0 & \text{for } x \neq 0, \\ \in \mathbb{R} & \text{for } x = 0. \end{cases}$$

and the shape of the Fourier transformed is (see Meier and Keller (1990, [18]))

$$\mathcal{F}\{f(x)\} = \frac{1}{\alpha^2 + \omega^2} \tag{5.5}$$

Because of the non-linearity of this function, a Taylor expansion of the above model has to be done

$$f(x) = f(x_0) + \left.\frac{\partial f}{\partial a_0}\right|_0 \Delta a_0 + \left.\frac{\partial f}{\partial a_1}\right|_0 \Delta a_1 + \left.\frac{\partial f}{\partial a_2}\right|_0 \Delta a_2 + \mathcal{O}(\Delta a^2) \tag{5.6}$$

neglecting derivatives of higher order $\mathcal{O}(\Delta a^2)$ for small Δa, with

$$a_0 = a_0^{(0)} + \Delta a_0 \qquad a_1 = a_1^{(0)} + \Delta a_1 \qquad a_0 = a_2^{(0)} + \Delta a_2$$

and with the partial derivatives at the given Taylor point of expansion

$$f(x_0) = a_0^{(0)} + a_1^{(0)} e^{a_2^{(0)} x} \qquad \left.\frac{\partial f}{\partial a_0}\right|_0 = 1 \qquad \left.\frac{\partial f}{\partial a_1}\right|_0 = e^{a_2^{(0)} x} \qquad \left.\frac{\partial f}{\partial a_2}\right|_0 = a_1^{(0)} x e^{a_2^{(0)} x}$$

To get approximate values of the unknown coefficients, the following assumptions have been done

$$a_0^{(0)} = 0 \tag{5.7}$$

$$a_1^{(0)} = \hat{C}(0) \tag{5.8}$$

because $a_0^{(0)}$ is in a linear context, it can be approximated with null, for $a_1^{(0)}$ we can say it is $\hat{C}(0)$ because $e^0 = 1$ and the parameter $a_2^{(0)}$

$$\begin{aligned} &\hat{C}(i) = a_1^{(0)} e^{a_2^{(0)} i} \\ \Leftrightarrow &\frac{\hat{C}(i)}{a_1^{(0)}} = e^{a_2^{(0)} i} \\ \Leftrightarrow &ln\left(\frac{\hat{C}(i)}{a_1^{(0)}}\right) = a_2^{(0)} i \end{aligned}$$

estimated by

$$a_2^{(0)} = \frac{1}{N} \sum_{i=0}^{N} \frac{ln(|\hat{C}(i)|/a_1^{(0)})}{i} \tag{5.9}$$

After that, an iterative least squares adjustment can be done.

The estimated parameters of the polynomial can be graphically interpreted as

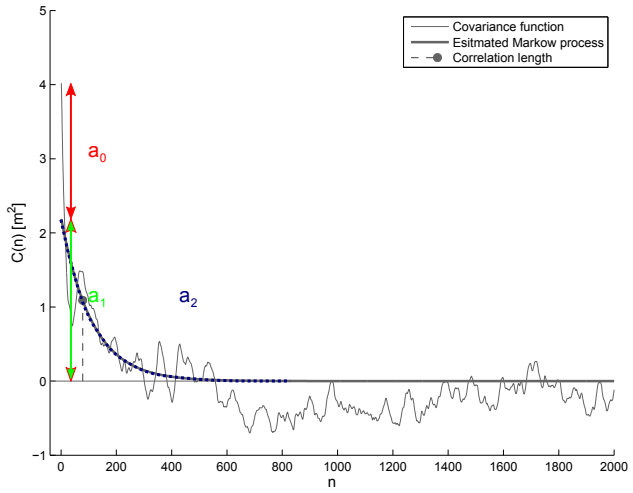


Figure 5.4: Graphical interpretation

the portion of white noise a_0, a Markov process with the variance a_1 and a parameter a_2, which describes the decay behavior of the polynomial and thus the correlation length of the signal.

5.1.2 Significance test

To get knowledge about the quality of the estimated coefficients, a significance test is applied after the estimation.

With the estimated residuals

$$\hat{e} = y - \hat{y} \tag{5.10}$$

we can compute the empirical standard deviation

$$s_0 = \sqrt{\frac{v^T v}{f}} \tag{5.11}$$

with $f = m - n$, m the number of observations and n the number of unknown parameters.

The standard deviation of the estimated parameters is

$$N^{-1} = (A^T A)^{-1} \tag{5.12}$$

$$S_{\hat{x}_i} = s_0 \sqrt{Q_{x_i x_i}} = s_0 \sqrt{tr(N^{-1})} \tag{5.13}$$

and the test value

$$t_i = \left| \frac{\hat{x}_i}{S_{\hat{x}_i}} \right| \sim t_f \tag{5.14}$$

which is t distributed, with f degrees of freedom.

Now we can formulate the test hypothesis

$$H_0 : t_i > t_{f,1-\alpha} \Rightarrow \text{Parameter is significant} \tag{5.15a}$$
$$H_a : t_i \leq t_{f,1-\alpha} \Rightarrow \text{Parameter is not significant} \tag{5.15b}$$

with a probability of $\alpha = 95°$.

5.1.3 Results

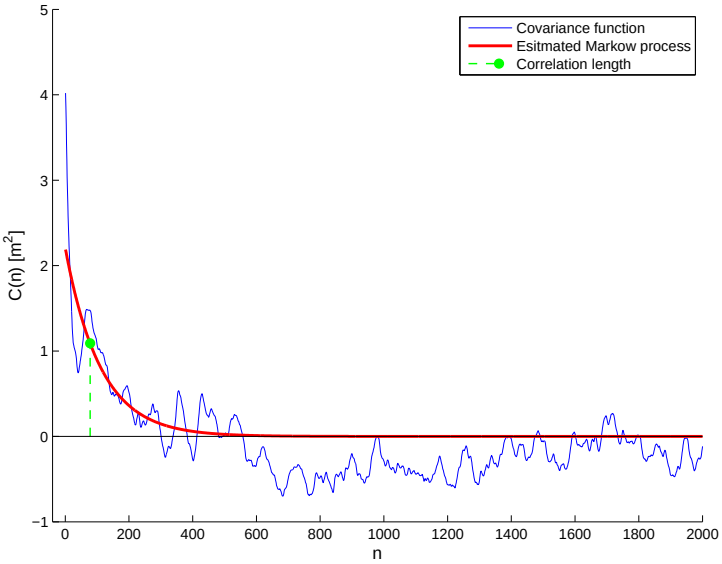


Figure 5.5: Covariance function with adjusted curve

The adjusted coefficients of the analyzed signals are summarized in the following table.

For GPS

Table 5.1: Estimated Markow process for GPS observations

Date	PRN	Frequency	$\hat{a}_0$	$\hat{a}_1$	$\hat{a}_2$	t_C [s]
August, 22	2	L1	~~-0.0006~~	1.4164	-0.0308	23
		L2-P(Y)	~~-0.0017~~	3.8421	-0.0308	23
	10	L1	~~-0.0154~~	1.9976	-0.0172	41
		L2-P(Y)	~~-0.0419~~	5.4185	-0.0172	41
August, 22	5	L1	0.7520	1.0101	-0.0314	23
		L2C	2.0399	2.7400	-0.0314	23
	17	L1	1.0462	0.5742	-0.0189	38
		L2C	2.8378	1.5575	-0.0189	38

for modernized GPS

Table 5.2: Estimated Markow process for modernized GPS observations

Date	PRN	Frequency	$\hat{a}_0$	$\hat{a}_1$	$\hat{a}_2$	t_C [s]
July, 25	1	L1	0.0898	0.3772	-0.0104	67
		L5	0.2890	1.2130	-0.0104	67
	25	L1	-0.0300	2.1228	-0.0135	52
		L5	-0.0967	6.8266	-0.0135	52

and for Galileo

Table 5.3: Estimated Markow process for Galileo observations

Date	PRN	Frequency	$\hat{a}_0$	$\hat{a}_1$	$\hat{a}_2$	t_C [s]
July, 25	101	L1bc	0.3097	1.4775	-0.0129	54
		E5a	0.9960	4.7516	-0.0129	54
	102	L1bc	0.4964	1.4765	-0.0119	59
		L5a	1.5966	4.7483	-0.0119	59

The correlation length t_C is approximately the time that elapses between $\hat{C}(0)$ and $\hat{C}(0)/2$. If a parameter is crossed out, it is not significant concerning the test in section 5.1.2.

5.1.4 Non-stationarity of ionospheric refraction

To get comparable results, a goal was to find related observations in the recorded data, where we can follow a satellite over the whole time, to get observations from minimal to maximal elevation and back to the horizon.

Some of these measurements show a non-stationarily behavior like the observation below.

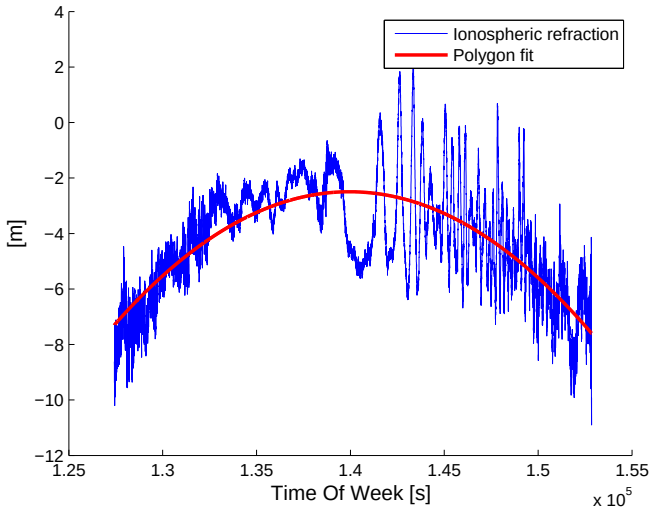


Figure 5.6: Non-stationarily behavior

The analysis, which is based on stationary processes can not be applied to this data. Because of this reason, the idea was to split the observation into two parts and analyze them separated. This behavior

was found in the data recorded on July 19th and will be analyzed separately from the other data, because it is not possible to remove those effects completely and thus, the results are not comparable to the data in 5.1.3.

For modernized GPS we get the following results

Table 5.4: Estimated Markow processes for non-stationary modernized GPS observations

Date	PRN	Frequency	$\hat{a}_0$	$\hat{a}_1$	$\hat{a}_2$	t_C [s]
July, 19	1	L1	0.1467	1.8437	-0.0068	102
		L5	0.4718	5.9291	-0.0068	102
	25	L1	~~0.0762~~	2.0811	-0.0045	153
			-1.1177	6.8224	-0.0178	40
		L5	~~0.2452~~	6.6926	-0.0045	153
			-3.5943	21.9397	-0.0178	40

and for Galileo

Table 5.5: Estimated Markow processes for non-stationary Galileo observations

Date	PRN	Frequency	$\hat{a}_0$	$\hat{a}_1$	$\hat{a}_2$	t_C [s]
July, 19	101	L1bc	0.8232	6.3089	-0.0086	81
		E5a	2.6472	20.2883	-0.0086	81
	102	L1bc	0.6180	1.6178	-0.0068	102
			~~0.0426~~	6.5962	-0.0093	75
		L5a	1.9873	5.2025	-0.0068	102
			~~0.1370~~	21.2122	-0.0093	75

It is still not clear, why these effects occur. Maybe they come from the receiver hardware, but this is not examined. It is strange, that these longwave effects begin to occur if the satellite is reaching maximum elevation.

5.2 Empirical power spectral density function

The empirical power spectral density function (PDF) $\hat{S}(\omega)$

$$S(\omega) = \mathcal{F}\{C\} \tag{5.16}$$

$$\hat{S}(\omega) = \mathcal{F}\{X_t\}(\mathcal{F}\{X_t\})^* = |\mathcal{F}\{X_t\}|^2 \tag{5.17}$$

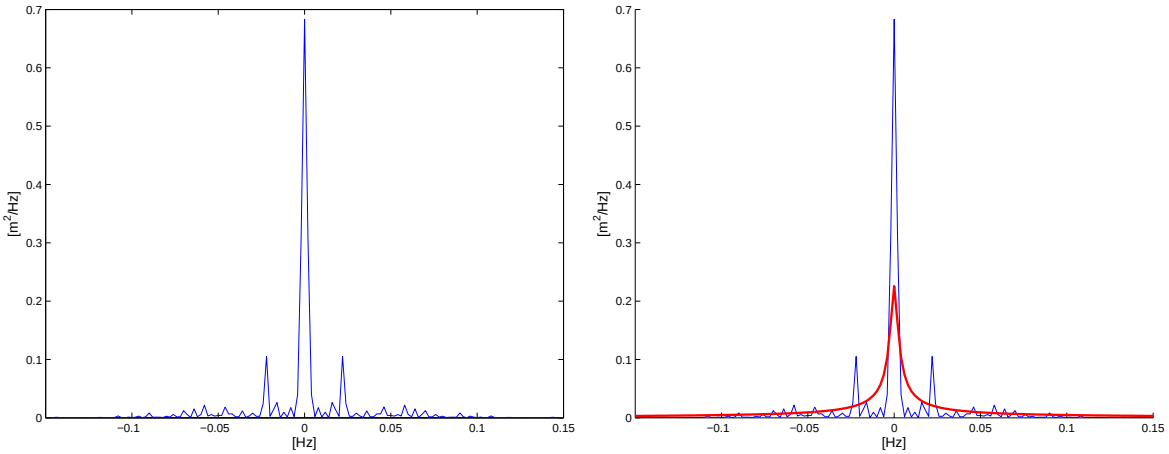


Figure 5.7: Power spectral density function

of the stationary process (left) and with the PDF of the estimated Markow process (right).

5.3 Conclusions

Looking at the results of the estimated covariance function for (modernized) GPS and Galileo we find the shortest correlation length for the signal of the conventional GPS. The correlation length of modernized GPS and Galileo can be interpreted as approximately the same length. It is varying a bit from satellite to satellite, this can be explained by the fact, that the elevation of the satellite was not the same over time and thus, the signal can be affected by different influences.

For the correlation length of the different frequencies, received at the same time with the same satellite, we can detect exactly the same length, which suggests that the frequency of the signal does not affect the correlation length.

Another interesting effect is the factor between the white noise a_0 (the same goes for colored noise a_1) on the different frequencies. It is for all measurements with the same frequency difference a constant factor $a_{i,f_2}/a_{i,f_1}$, which is the same for white noise a_0 and colored noise a_1.

Table 5.6: Constant factor between estimated noise behavior

System	Frequency f_1	Frequency f_2	Difference [MHz]	Factor $a_{i,f_2}/a_{i,f_1}$	f_1^2/f_2^2
GPS	L1	L2	347.82	2.7	1.28
GPS	L1	L5	398.97	3.2	1.34
Galileo	L1bc	E5a	399.97	3.2	1.34

The factor depends obviously on the difference of the frequencies received. The influence of ionospheric refraction and tropospheric refraction does also depend on the frequency, the signal is transmitted with. So the idea was to check if this effect is based on refraction, which can be explained by f_1^2/f_2^2. Looking at the results, we con not say, that this effect is purely depended on refraction. Maybe it is possible that there is a noise amplification in the hardware of the receiver, which leads to this effect.

The results of the power spectral density function (PDF) of the received signal and the PDF of the estimated Markow process confirm the applied estimation. At 0 Hz we can see the superposition of white noise and colored noise again. Sometimes we can find sidelobes in the spectrum which may occur if there

are still longwave effects left in the signal. This should not happen, but if we look at the effects described in section 5.1.4 there may still be small effects left in the signal, which do not affect the analysis in a strong way, but can be noticed analyzing the PDF.

For this kind of process, the applied covariance model

$$f(x) = a_0 + a_1 e^{a_2 x} \tag{5.18}$$

is not sufficient. For a better approximation a model like

$$g(\tau) = e^{-\tau\alpha} \cos \tau\beta \tag{5.19}$$

with the Fourier transform (see Meier and Keller (1990, [18]))

$$\mathcal{F}\{g(\tau)\} = \frac{1}{\alpha^2(t \pm \beta)^2} \tag{5.20}$$

should be used. With this model, the occurring sidelobes in the PDF will be considered.

To sum up the results, it was possible to analyze the received GNSS data, for conventional GPS, for modernized GPS and for the upcoming Galileo Global Navigation Satellite Systems, with the restriction, that there are only two Galileo test satellites in orbit, at this time. To get a signal, which can be analyzed, it was necessary to remove receiver specific effects from the signal and to create stationary processes. As described above, there are still open questions concerning the receiver with its hard- and software components. In spite of everything, it was also possible to find credible observations and to determine their correlation length and their noise behavior, which was the main goal of this thesis.

References

[1] Beidou navigation system. Website, 2010. Available online at `http://en.wikipedia.org/wiki/Beidou_navigation_system` visited on September 19th 2010.

[2] Binary offset carrier. Website, 2010. Available online at `http://en.wikipedia.org/wiki/Binary_offset_carrier` visited on September 19th 2010.

[3] Galileo (satellite navigation). Website, 2010. Available online at `http://en.wikipedia.org/wiki/Galileo_(satellite_navigation)` visited on September 19th 2010.

[4] Global positioning system. Website, 2010. Available online at `http://en.wikipedia.org/wiki/Global_Positioning_System` visited on September 19th 2010.

[5] Glonass. Website, 2010. Available online at `http://en.wikipedia.org/wiki/GLONASS` visited on September 19th 2010.

[6] Norad two-line element set format. Website, 2010. Available online at `http://celestrak.com/NORAD/documentation/tle-fmt.asp` visited on September 19th 2010.

[7] Phase-shift keying. Website, 2010. Available online at `http://en.wikipedia.org/wiki/Phase-shift_keying` visited on September 19th 2010.

[8] J. J. Spilker Jr. Global Positioning System: Theory and Application, volume 1, chapter 3, pages 66–79. American Institute of Aeronautics and Astronautics, 1996.

[9] J. J. Spilker Jr. Global Positioning System: Theory and Application, volume 1, chapter 13. American Institute of Aeronautics and Astronautics, 1996.

[10] Wolfgang Keller. Observation techniques in satellite geodesy, March 2007.

[11] Wolfgang Keller. Grundlagen der satellitengeodäsie, January 2009.

[12] J. A. Klobuchar. Global Positioning System: Theory and Application, volume 1, chapter 12. American Institute of Aeronautics and Astronautics, 1996.

[13] Navstar GPS Joint Program Office, USA, El Segundo, CA. Interface Specification, IS-GPS-200, revision d edition, December 2004.

[14] Bradford W. Parkinson. Global Positioning System: Theory and Application, volume 1, chapter 1, page 3. American Institute of Aeronautics and Astronautics, 1996.

[15] Bradford W. Parkinson. Global Positioning System: Theory and Application, volume 1, chapter 11. American Institute of Aeronautics and Astronautics, 1996.

[16] Septentrio. SBF Reference Guide. Spetentrio Satellite Navigation, B-3001 Leuven, Belgium, 1.8.0 edition, December 2009.

[17] Nico Sneeuw. Dynamic satellite geodesy, october 2006.

[18] Siegfried Meier und Wolfgang Keller. Geostatistik. Einführung in die Theorie der Zufallsprozesse. Springer, 1990.

A Source code

The following chapter contains the source code and a description of all functions, used for all calculations, sorted by chapters.

A.1 Signal conversion

The main program of the signal conversion part converts the *.sbf files from Septentrio into *.mat files, which can be easily used for further applications. First of all one has to define the input files and the file names an locations to which the *.mat files should be written. Reading the data into *.mat files is done by the function readData.m.

To get only the data, which is recorded simultaneously on each frequency for each satellite, it is necessary to synchronize the measured data (done in the syncEpoch.m function). Because of the huge measurement files, which have to be cached into the RAM (Random Access Memory) of the computer if they should be used for calculations, Matlab runs out of Memory. To avoid this failure, the main program splits the recorded data into pieces, computes them and afterwards fit's them together. After the measurement epochs are synchronized with the function `syncEpoch.m` the file size decreases and can be used without splitting operations in future.

Because of the receiver clock drift, the measured pseudorange has to be corrected by the time lag referring GNSS time. This is done by the `removeClockBias.m` function.

The functions `plausibilityCheck.m` and `visualisation_stair_elimination.m` provide routines, to test the manipulated data and verify the correctness.

```
1  clear all
2  close all
3  clc
4  format long g
5
6  tic
7
8  path = '../../00_Data/';
9  file = 'AAU_SEPT2340.10';
10
11 inFileMeas = strcat(path, file, '__MeasEpoch_2.stf');
12 outFileMeas = strcat(path, file, '__MeasEpoch1_raw.mat');
13 % outFileMeas = 'measEpoch1.mat';
14
15 % inFilePVT = strcat(path, file, '__PVTGeodetic_2.stf');
16 % outFilePVT = strcat(path, file, '__PVTGeodetic.mat');
17 inFilePVT = strcat(path, file, '__PVTCartesian_2.stf');
18 outFilePVT = strcat(path, file, '__PVTGeodetic.mat');
19
20 inFileAlm = strcat(path, file, '__GPSAlm_1.stf');
21 outFileAlm = strcat(path, file, '__GPSAlm.mat');
22
23 %%% converting stf files to mat files
24 disp('Converting stf to mat files')
25
26 % readData(inFileMeas, outFileMeas, 'measEpoch');
27 readData(inFilePVT, outFilePVT, 'PVTGeodetic');
28 readData(inFileAlm, outFileAlm, 'GPSAlm');
29
30 disp('Loading relevant mat files')
31 %%% load measEpoch
32 load(outFileMeas)
33
34 %%load PVTGeodetic v2
35 load(outFilePVT)
36
```

```
37 %%% measEpoch column discription
38 % column 8: pseudorange [m] from c/a code
39 % column 9: pseudorange [m] from phase measurement
40
41 %%% PVTGeodetic column discription
42 % column 1 : WNc [week}
43 % column 2 : TOW [s]
44 % column 13: ClockBias [ms]
45
46
47 % if measEpoch is too large, split it into pieces of xxx rows, after
48 % convertion, merge data and save
49
50 disp('Processing data')
51
52 maxSize = 200000;
53 SizeMeasEpoch = size(measEpoch,1);
54
55
56 numParts = ceil(SizeMeasEpoch / maxSize);
57 measEpochNew = [];
58
59 for part=1:numParts
60     tic
61
62     % split data
63     if(part == 1)
64         data{part} = measEpoch(1+((part-1)*maxSize):part*maxSize-1 ,:);
65     elseif( part > 1 && part < numParts)
66         data{part} = measEpoch((part-1)*maxSize:part*maxSize-1 ,:);
67     elseif (part == numParts)
68         data{part} = measEpoch((part-1)*maxSize:end ,:);
69     end
70
71     %%% syncEpoch
72     data{part} = syncEpoch(data{part});
73
74     measEpochNew = [measEpochNew; data{part}];
75
76     time = toc;
77     disp(['Time to sync part ',num2str(part),' of ',num2str(numParts),': ', num2str(time)])
78
79 end
80 measEpoch = measEpochNew;
81 clear measEpochNew
82
83 measEpoch = sortrows(measEpoch,2);
84
85 disp('Removing doubled entries...')
86 measEpoch = remDoubleEntries(measEpoch);
87
88
89 disp('Removing clock bias...')
90 [measEpoch] = removeClockBias(measEpoch, PVTGeodetic);
91
92
93 disp('Saving processed data')
94 save(strcat(path, file, '__MeasEpoch1_processed.mat'), 'measEpoch')
95
96 disp('Processing finished!!!')
```

A.1.1 Functions and test routines

readData.m

```
%%% read data in mat file
% after loading mat file, content is stored in variable data
% inFile: Path of input data as string
% outFile: Path of output data as string
% Example:
% inFile = 'data/STGT1230.10_PVTGeodetic_2_5secInterval.stf';
% outFile = 'data/STGT1230.10_PVTGeodetic_2_5secInterval.mat';
% DataName = 'PVTGeodetic'

function readData(inFile, outFile, DataName)

%%% read data and save as *.mat file
var = genvarname(DataName);
data = csvread(inFile);
eval([var ' = data;']);
save(outFile, DataName);
```

syncEpoch.m

```
%%% synchronize epoch (TOW) between different frequencies

function [measEpoch] = syncEpoch(measEpoch)

measEpochNew = [];

for satNumber=1:106

    % find all data of sattelite X
%     posSatX = find( measEpoch(:,4) == satNumber);
%     dataSatX = measEpoch(posSatX,:);
%     dataSatX(:,15) = posSatX;

    dataSatX = measEpoch(measEpoch(:,4) == satNumber, :);


    % browse all frequencies
    % get freq index and number of measurements
    for freq=0:22

        freqLength(freq+1, 1) = freq;
        freqLength(freq+1, 2) = length( dataSatX( dataSatX(:,7) == freq, : ) );

    end
    freqLength = sortrows(freqLength,2);
    freqLength( freqLength(:,2) == 0 ,:) = [];

    % if measurements for sat x are available, start comparison
    if(¬isempty(freqLength))

%     measEpochNew = [measEpochNew; dataSatX( dataSatX(:,7) == freqLength(1,1), : )];

% size(freqLength,1)

        for i=2:size(freqLength,1)

            data1 = dataSatX( dataSatX(:,7) == freqLength(1,1), : );
            data2 = dataSatX( dataSatX(:,7) == freqLength(i,1), : );

            % if size of data1==size of data2
            if(size(data1,1) == size(data2,1))

                data1(data1(:, 2) ≠ data2(:, 2) , :) = [];
```

```
44
45              end
46
47
48              if(size(data1,1) ≠ size(data2,1))
49
50                  % compare epoch
51                  deleted = 1;
52                  while(deleted == 1)
53
54                      % data1 has to be smaller than data2
55                      if( size(data1,1) > size(data2,1) )
56                          dummy = data1;
57                          data1 = data2;
58                          data2 = dummy;
59                          clear dummy
60                      end
61
62                      l1 = size(data1,1);
63
64                      dataLeft = data1;
65                      dataRight = data2(1:l1, :);
66
67
68                      % find position of unequal epoch
69                      pos = find( dataLeft(:,2) ≠ dataRight(:,2) );
70
71                      if( ¬isempty(pos) )
72
73                          posEpoch =  data2(:, 2) == dataRight(pos(1), 2) ;
74                          % delete unequal epoch in data2
75                          data2(posEpoch, :) = [];
76
77                      else
78                          deleted = 0;
79                      end
80
81
82                  end
83
84                  % delete epoch at the end of data2 which is not in data1
85                  if(length(data1) < length(data2))
86                      data2 = data2(1:length(data1),:);
87                  end
88
89              end
90
91                   % write compared data into measEpochNew and clear variables
92                   measEpochNew = [measEpochNew; data1];
93                   measEpochNew = [measEpochNew; data2];
94                   clear data1
95                   clear data2
96
97  %              end
98
99
100
101         end
102
103
104     end
105
106
107 end
108
109 measEpoch = measEpochNew;
```

removeClockBias.m

```
function [measEpoch] = removeClockBias(measEpoch, PVTGeodetic)

%%% output
% code [m]
% phase [m]

%%% PVTGeodetic column discription
% column 1 : WNc [week}
% column 2 : TOW [s]
% column 13: ClockBias [ms]

%%%%%% removeClockBias

freqCatalog         = zeros(31,1);
freqCatalog(1:2)    = 1575.42;
freqCatalog(3:4)    = 1227.60;
freqCatalog(5)      = 1176.45;
freqCatalog(17:18)  = 1575.42;
freqCatalog(19:20)  = 1278.75;
freqCatalog(21)     = 1176.45;
freqCatalog(22)     = 1207.14;
freqCatalog(23)     = 1191.795;

c = 2.99792458e8;

lengthMeas = size(measEpoch,1);
percentDisp = 0;
posPVTbefore = 0;
for i=1:lengthMeas

    posPVT = find( measEpoch(i,2) == PVTGeodetic(:,2) );

    % interpolate clock bias if there is no value in PVTGeodetic available
    if( isempty(posPVT) )
        delay = interp1(PVTGeodetic(posPVTbefore-2:posPVTbefore+2, 2), ...
                    PVTGeodetic(posPVTbefore-2:posPVTbefore+2, 13), ...
                    measEpoch(i,2));
    else
        delay = PVTGeodetic(posPVT,13);
        posPVTbefore = posPVT;
    end

    % pseudorange correction
    dp = c * delay * 10^-3;

    % update pseudorange
    measEpoch(i,8) = measEpoch(i,8) - dp;

    % update carrier
    if( measEpoch(i, 9) ~= 0 )
    freq = freqCatalog( measEpoch(i, 7) + 1 ) * 10^6;
    lambda = c / freq;

    measEpoch(i, 9) = measEpoch(i, 9)*lambda - dp;
    end

    % display status
    percent = uint8(i/lengthMeas*100);
    if(percent > percentDisp)
        disp([num2str(percent), '% finished']);
        percentDisp = percent;
    end

end
```

plausibilityCheck.m

```
clear all
close all
clc
format long g

%%% plausibility check

%%% load measEpoch
% load data/STGT1230.10_MeasEpoch_5secInterval_jumps_removed_syncEpoch.mat
% measEpochRed = measEpoch;
% load data/STGT1230.10_MeasEpoch_5secInterval_syncEpoch.mat
% measEpochOrig = measEpoch;
% clear measEpoch

load ../../00_Data/LMU_STGT2000.10__MeasEpoch_raw.mat
measEpoch_raw = measEpoch;

load ../../00_Data/LMU_STGT2000.10__MeasEpoch_processed.mat

%%% get measEpoch of sat #
% 1 - 37: PRN of GPS
% 71 - 106: PRN of GALILEO with offset of 70
PRN = 25;
satRed = measEpoch ( measEpoch(:,4) == PRN ,: );
satOrig = measEpoch_raw ( measEpoch_raw(:,4) == PRN ,: );

%%% filter signal type
% signal number row 7 in measEpoch
% 0  GPS_L1-CA   1575.42
% 1  GPS_L1-P(Y) 1575.42
% 2  GPS_L2-P(Y) 1227.60
% 3  GPS_L2C     1227.60
% 4  GPS_L5      1176.45
% 16 GAL_L1A     1575.42
% 17 GAL_L1BC    1575.42
% 20 GAL_E5a     1175.45
% 21 GAL_E5b     1207.14
% 22 GAL_E5      1191.795

satF1Red = satRed ( satRed(:,7) == 0  , : );
satF2Red = satRed ( satRed(:,7) == 4 , : );
satF1Orig = satOrig ( satOrig(:,7) == 0  , : );
satF2Orig = satOrig ( satOrig(:,7) == 4 , : );

diffRedPseudorange = satF1Red(1:25000,8) - satF2Red(1:25000,8);
diffOrigPseudorange = satF1Orig(2:25001,8) - satF2Orig(1:25000,8);

%%% reduced measurements (Jumps and sync)
figure
subplot(3,1,1)
plot(satF1Red(:,2), satF1Red(:,8))
title('Reduced measurements \rho_{f1}', 'Fontsize',12)

subplot(3,1,2)
plot(satF2Red(:,2), satF2Red(:,8))
title('\rho_{f2}', 'Fontsize',12)

subplot(3,1,3)
plot(satF2Red(1:25000,2), diffRedPseudorange)
title('\rho_{f1} - \rho_{f2}', 'Fontsize',12)
xlabel('Time Of Week [s]', 'Fontsize',12)

print -depsc plots/plausibilityCheck/reducedMeasurements

%%% original measurements just synchonized
figure
subplot(3,1,1)
plot(satF1Orig(:,2), satF1Orig(:,8))
title('Original measurements \rho_{f1}', 'Fontsize',12)
```

```
70
71 subplot(3,1,2)
72 plot(satF2Orig(:,2), satF2Orig(:,8))
73 title('Pseudorange \rho_{f2}', 'Fontsize',12)
74
75 subplot(3,1,3)
76 plot(satF2Orig(1:25000,2), diffOrigPseudorange)
77 title('\rho_{f1} - \rho_{f2}', 'Fontsize',12)
78 xlabel('Time Of Week [s]', 'Fontsize',12)
79
80 print -depsc plots/plausibilityCheck/originalMeasurements
81
82
83 figure
84 plot(satF2Orig(1:25000,2), diffRedPseudorange - diffOrigPseudorange)
85 xlabel('Time Of Week [s]', 'Fontsize',12)
86 ylabel('Pseudorange [m]', 'Fontsize',12)
87
88 print -depsc plots/plausibilityCheck/difference
```

visualisation_stair_elimination.m

```
1  clear all
2  close all
3  clc
4  format long g
5
6
7  %%% load measEpoch
8  % load data/STGT1230.10_MeasEpoch_5secInterval.mat
9  % measEpoch = data;
10 % clear data
11
12
13 %%% measEpoch column discription
14 % column 8: pseudorange [m] from c/a code
15 % column 9: pseudorange [cycles] from phase measurement
16
17 %%%load PVTGeodetic v2
18 % load data/STGT1230.10_PVTGeodetic_2_5secInterval.mat
19 % PVTGeodetic = data;
20 % clear data
21
22 %%% PVTGeodetic column discription
23 % column 1 : WNc [week}
24 % column 2 : TOW [s]
25 % column 13: ClockBias [ms]
26
27 %%% vis clock bias
28
29 load ../../00_Data/LMU_STGT2000.10__MeasEpoch_raw.mat
30 measEpoch_raw = measEpoch;
31
32 load ../../00_Data/LMU_STGT2000.10__MeasEpoch_processed.mat
33
34 PRN = 25;
35
36 sat_raw =  measEpoch_raw ( measEpoch_raw(:,4) == PRN ,: );
37 sat = measEpoch ( measEpoch(:,4) == PRN ,: );
38
39 figure
40 hold on
41 plot(sat_raw(1000:10000,2), sat_raw(1000:10000,8))
42 xlabel('Time Of Week [s]', 'Fontsize',12)
43 ylabel('Pseudorange [m]', 'Fontsize',12)
44 print('-depsc', 'plots/staircase')
45
```

```
46 figure
47 hold on
48 plot(sat_raw(:,2), sat_raw(:,8),'r')
49 plot(sat(:,2), sat(:,8), 'LineWidth', 2)
50 xlabel('Time Of Week [s]', 'Fontsize',12)
51 ylabel('Pseudorange [m]', 'Fontsize',12)
52 print('-depsc', 'plots/visualisation_removeJumps')
53
54 figure
55 plot(sat(:,2), sat(:,8))
56 xlabel('Time Of Week [s]', 'Fontsize',12)
57 ylabel('Pseudorange [m]', 'Fontsize',12)
58 title('Code observation', 'Fontsize',12)
59 print('-depsc', 'plots/visualisation_rho')
60
61 figure
62 plot(sat(:,2), sat(:,9))
63 xlabel('Time Of Week [s]', 'Fontsize',12)
64 ylabel('Pseudorange [m]', 'Fontsize',12)
65 title('Phase observation', 'Fontsize',12)
66 print('-depsc', 'plots/visualisation_phi')
67
68 %%% vis equalisation
69
70 data = [2.2 2 2 1.8]*10^7;
71 time = [1.33 1.34 1.342 1.352]*10^5;
72
73 figure
74 hold on
75 plot(time, data)
76 xlabel('Time Of Week [s]', 'Fontsize',12)
77 ylabel('Pseudorange [m]', 'Fontsize',12)
78 print('-depsc', 'plots/visualisation_equalisation')
```

A.2 Elevation

For further analysis of the signal, the elevation can provide useful information to check, if there is still a geometrical component in the signal. Thales corporation was also interested to get the elevation for their own investigations. Thus the elevation will be calculated with the routine `elevation.m`. It calculates the elevation using the almanac information the satellite transmits to the receiver. This is working for most of the GPS satellites, Galileo satellites do not transmit almanac information at this time. To be correct, maybe they do transmit them, but the receiver is not able to log them.

Therefore we tried to use TLE orbital parameters. They are available for all kinds of objects in space. Because it is difficult or boring to assign the TLE file which is updated several times a week to the current GNSS observation, the function `readTLEorbitalElements.m` reads out all TLE files of a folder, transfers them into the almanac format and gives them back to the main elevation function.

For GPS satellites, a so called YUMA almanac is available. It is more or less the same as TLE, but does not have to be transferred into the almanac format as TLE has to be. The function `readYUMAAlmanac.m` does read the YUMA files.

Having the almanac information it is necessary to assign the latest almanac information to the current observation. This is done in the `processElevation.m` function, which also gives the matching parameters to the `Alm2xyz.m` function, which calculates the Earth-fixed satellite coordinates and gives after that, together with the receiver antenna coordinates, the data to the `posRecSat2el.m` function, which calculated the elevation.

```
clear all
close all
format long g
clc

%%% compute elevation of each satellite %%%

%%% additional columns to measEpoch file
% 15-17 x,y,z earth-fixed coordinates [m]
% 18 elevation [rad]

%%% file definitions

disp('Start reading input files')

path = '../../../00_Data/';
file = 'AAU_SEPT2340.10';

load(strcat(path, file, '__GPSAlm.mat'));
load(strcat(path, file, '__MeasEpoch_processed.mat'));

TLEfolder = 'almanac/galileo_tle/';

%%% receiver antenna position
% until 20. july
% xR = [4154168.898172 666191.330124008 4778133.24835098]';

% from 20. july
xR = [4154187.15723352 666194.258273724 4778117.64032416]';


%%% GPSAlm column definition
% 1 WNc[week]
% 2 TOW[s]
% 3 PRN
% 4 E
% 5 t_oa[s]
```

```
38  % 6 Delta_i[semi-circle]
39  % 7 OMEGADOT[semi-circle/s]
40  % 8 SQRT_A[m^0.5]
41  % 9 OMEGA_0[semi-circle]
42  % 10 omega[semi-circle]
43  % 11 M_0[semi-circle]
44  % 12 a_f1[s/s]
45  % 13 a_f0[s]
46  % 14 WN_a[week]
47  % 15 AS
48  % 16 health8
49  % 17 health6
50
51
52  %%% read YUMA format not used right now...!
53
54  % file = 'almanac/gps_yuma/082.alm';
55  %
56  % [alm] = readYUMAAlmanac(file)
57
58  %%% end read YUMA format
59
60
61  %%% read TLE format
62
63  disp('Reading TLE files')
64
65  [Alm] = readTLEorbitalElements(TLEfolder);
66
67  GPSAlm = [GPSAlm; Alm];
68
69  %%% end TLE format
70
71
72
73  %%% compute satellite position, earth fixed, ref icd-200 p. 99
74
75  disp('Processing satellite positions and elevation')
76  [measEpoch] = processElevation(measEpoch, GPSAlm, xR);
77
78  disp('Saving output file')
79  save(strcat(path, file, '__MeasEpoch_processed.mat'),'measEpoch')
80
81  disp('Finished!')
82
83
84  %%% tests
85
86  % choose PRN
87  PRN = 102;
88
89  posInMeasEpoch = find(measEpoch(:,4) == PRN);
90  data = measEpoch(posInMeasEpoch,:);
91
92  t = data(:,2);
93
94
95  xk = data(:,15:17);
96
97
98  % gamma = Omegak';
99  %
100 % X = R3(gamma, xk');
101
102 % GPSAlm(:,18:20) = [xk yk zk];
103
104 % sat = GPSAlm(GPSAlm(:,3) == 1,:);
105
106 % x =sat(:,18);
```

```
107 % y =sat(:,19);
108 % z =sat(:,20);
109
110 figure
111 hold on
112 plot3(xk(:,1),xk(:,2),xk(:,3),'x')
113 % plot3(X(1,:),X(2,:),X(3,:),'rx')
114 % plot3(x,y,z,'ro')
115 axis equal
116
117 % earth
118 R = 6371000;
119 load coast.mat
120 hold on
121 [lambda,phi] = meshgrid(-pi:pi/100:pi,-pi/2:pi/100:pi/2);
122 [x,y,z] = sph2cart(lambda,phi,R);
123 surf(x,y,z)
124 colormap([.8,.8,.8])
125 shading interp
126 [x,y,z] = sph2cart(long*pi/180,lat*pi/180,R);
127 plot3(x,y,z,'Color','black');
128 % end earth
129
130 plot3(xR(1),xR(2),xR(3),'gx')
131
132 % neg = find(el < 0);
133 % plot3(xk(neg,1),xk(neg,2),xk(neg,3),'gx')
134
135
136 figure
137 plot(t,data(:,8))
138 title('pseudorange')
139 %
140 figure
141 hold on
142 plot(t,data(:,18)*180/pi)
143 % title('elevation')
144 xlabel('Time Of Week [s]', 'Fontsize',12)
145 ylabel('Elevation [°]', 'Fontsize',12)
146 % print('-depsc', 'plots/elevation')
```

A.2.1 Functions and test routines

processElevation.m

```
1  % Process elevation computation for each satellite included in the
2  % measEpoch file
3  % Input:    measEpoch
4  %           Alm (Almanac)
5  %           xR Receiver antenna position [x,y,z]' (earth fixed)
6  % Output:   measEpoch with additional columns 15-17 [x y z] coordinates of
7  %           each satellite relating to earth fixed coordinate system [m]
8  %           and column 18 with elevation [rad]
9
10 %%% Alm column definition
11 % 1 WNc[week]
12 % 2 TOW[s]: -99 indicates tle (galileo), no information available
13 % 3 PRN
14 % 4 E
15 % 5 t_oa[s]
16 % 6 Delta_i[semi-circle]
17 % 7 OMEGADOT[semi-circle/s]
18 % 8 SQRT_A[m^0.5]
19 % 9 OMEGA_0[semi-circle]
20 % 10 omega[semi-circle]
21 % 11 M_0[semi-circle]
22 % 12 a_f1[s/s]
```

```
23  % 13 a_f0[s]
24  % 14 WN_a[week]
25  % 15 AS
26  % 16 health8
27  % 17 health6
28
29  function [measEpoch] = processElevation(measEpoch, GNSSAlm, xR)
30
31
32  for PRN=1:106
33
34
35
36      % get position of data of Sat PRN in measEpoch and fill this data into data
37      % variable
38      posInMeasEpoch = find(measEpoch(:,4) == PRN);
39      data = measEpoch(posInMeasEpoch,:);
40
41      if(¬isempty(data)) disp(['Processing Satellite ',num2str(PRN),'.']); end
42
43      % get almanac for prn
44      Alm = GNSSAlm(GNSSAlm(:,3) == PRN,:);
45      if(isempty(Alm) && ¬isempty(data))
46          disp(['No almanac information for Satellite ',num2str(PRN),' available.']);
47      end
48
49      if(¬isempty(Alm) && ¬isempty(data))
50          % find alamac data for the same week
51          Alm = Alm(Alm(:,1) == data(1,1),:);
52          if(isempty(Alm)) disp(['No almanac information for week ',...
53                  num2str(data(1,1)),' available.']);end
54      end
55
56      for intt0=1:size(Alm,1)
57
58          % find out, at which positions which almanac data has to be used
59          if( size(Alm,1) == 1 )
60              [xS] = Alm2xyz(Alm(intt0,:), data);
61              [el] = posRecSat2el(xR,xS);
62              measEpoch(posInMeasEpoch, 15:17) = xS;
63              measEpoch(posInMeasEpoch, 18) = el;
64          else
65              posInData = find( data(:,2) < Alm(intt0,5) );
66
67              [xS] = Alm2xyz(Alm(intt0,:), data(posInData, :));
68              [el] = posRecSat2el(xR,xS);
69              measEpoch(posInMeasEpoch(posInData), 15:17) = xS;
70              measEpoch(posInMeasEpoch(posInData), 18) = el;
71
72              % remove processed data from data
73              data(posInData,:) = [];
74          end
75
76          % if there is still data in data, then use latest t_oe for all t
77          % (could happen, if all t > t_oa )
78          if( ¬isempty(data) )
79              [c i] = max(Alm(:,5));
80              [xS] = Alm2xyz( Alm(i,:) , data);
81              [el] = posRecSat2el(xR,xS);
82              measEpoch(posInMeasEpoch, 15:17) = xS;
83              measEpoch(posInMeasEpoch, 18) = el;
84          end
85
86
87      end
88
89  end
```

readYUMAAlmanac.m

```
% read out alamnac parameters of GPS in YUMA format
% almanach (alm) structure
% row discription
% 1 ID
% 2 Health
% 3 e
% 4 t
% 5 i
% 6 OMEGADot
% 7 sqrt(a)
% 8 OMEGA
% 9 omega
% 10 M0
% 11 Af0
% 12 Af1
% 13 week

function [alm] = readYUMAAlmanac(file)

fid = fopen(file);

alm = [];
i = 1;
j = 1;

while ¬feof(fid)

    line = fgetl(fid);
    if ¬isempty(line)

        string = textscan(line, '%s %f','delimiter',':');

        if ¬isempty(string{2})
            alm(i,j) = string{2};

            i = i+1;
        end

    else
        j = j + 1;
        i = 1;
    end

end
```

readTLEorbitalElements.m

```
% read out orbital elements of Galileo in TLE Format
% almanach (alm) structure
% row discription
% 1 WNc[week]
% 2 TOW[s]: −99 indicates tle (galileo), no information available
% 3 PRN
% 4 E
% 5 t_oa[s]
% 6 Delta_i[semi−circle]
% 7 OMEGADOT[semi−circle/s]
% 8 SQRT_A[m^0.5]
% 9 OMEGA_0[semi−circle]
% 10 omega[semi−circle]
% 11 M_0[semi−circle]
% 12 a_f1[s/s]
% 13 a_f0[s]
```

```
% 14 WN_a[week]
% 15 AS
% 16 health8
% 17 health6

function [oe] = readTLEorbitalElements(folder)

s = dir(folder);
sday = 86400;


oe = zeros(1,17);
% for each file in directory 'folder'
for i=1:size(s,1)

    fid = fopen([ folder, s(i).name]);


    if fid > 0

    % read the file to the end

    while ~feof(fid)

        A = strtrim( fscanf(fid,'%13c%*s',1) );
        B = fscanf(fid,'%d%6d%*c%5d%*3c%2d%f%f%5d%*c%*d%5d%*c%*d%d%5d',[1,10]);
        C = fscanf(fid,'%d%6d%f%f%f%f%f%f%f',[1,9]);

        if( strcmp(A, 'GIOVE-A') )
            prn = 101;
        elseif ( strcmp(A, 'GIOVE-B') )
            prn = 102;
        end

        if( ~isempty(A) && ~isempty(B) && ~isempty(C) )

            mu = 3.986005e14; % [m^3/s^2]
            yr = B(1,4);
            epoch = B(1,5);

            di = C(1,3)/180-0.3; % DeltaInclination
            Om = C(1,4)/180; % Right Ascension of the Ascending Node
            e = C(1,5)/1e7; % Eccentricity
            om = C(1,6)/180; % Argument of perigee
            M = C(1,7)/180; % Mean anomaly
            n = C(1,8)*2*pi/(24*3600); % Mean motion
            % Calculate the semi-major axis
            sqrt_a = sqrt((mu/n.^2).^(1/3));


            % compute WNc and t_oa (compatible to GPSAlm)

            year = yr + 2000;

            % day = floor(epoch);
            day = datenum(year,0,0) + floor(epoch);

            % time of day [s]
            t = (epoch-floor(epoch))*sday;

            % number of days since 1980
            numDays = day - datenum(1980,01,6);

            % week
            WNc = floor( numDays / 7 );

            % day of week (0=sunday, 1=monday, ...)
            dow = weekday(day)-1;

```

```
86              % upload time, since week WNc
87              t_oa = floor(dow*sday + t);
88
89              % output
90              % if extry exists already, do not insert it again
91
92              if( isempty( find( t_oa == oe(WNc == oe(:,1), 5), 1 ) ) )
93                  oe = [oe; WNc -99 prn e t_oa di 0 sqrt_a Om om M 0 0 0 0 0 0];
94              end
95          end
96
97      end
98      fclose(fid);
99
100     end
101
102 end
103 oe(1,:) = [];
```

m2e.m

```
1  function E_neu = m2e(M,e)
2  % calculate mean anomaly from eccentical anomaly
3
4  E_alt = M;
5  E_neu = M+1;
6
7  while true
8   E_neu = e .* sin( E_alt ) + M;
9
10  if  sum ( 1 - ( abs(E_neu - E_alt) < 10^-10 ) )  == 0
11     break;
12  else
13    E_alt = E_neu;
14  end
15 end
```

ell2kart.m

```
1  % coordinate transformation: ellipsoidal -> cartesian
2
3  function x = ell2kart(a, e, H, B, L)
4
5  N = a./sqrt(1-e^2.*sin(B).^2);
6
7  x(1) = (N+H).*cos(B).*cos(L);
8  x(2) = (N+H).*cos(B).*sin(L);
9  x(3) = (N.*(1-e^2)+H).*sin(B);
```

Alm2xyz.m

```
1
2  % returns satellite positions in earth fixes coordinate system
3
4  function [xk] = Alm2xyz(Alm, data)
5
6  t = data(:,2);
7
8  mu = 3.986005e14; % [m^3/s^2]
9  OmegaEDot = 7.2921151467e-5; % [rad/s]
10 i0 = 0.3*pi;
11 e = Alm(:,4);
```

```
12 omega = Alm(:,10)*pi;
13 di = Alm(:,6)*pi;
14 Omega0 = Alm(:,9)*pi;
15 OmegaDot = Alm(:,7)*pi;
16 M0 = Alm(:,11)*pi;
17 toa = Alm(:,5);
18
19
20 % set parameters
21 A = Alm(:,8).^2;
22
23 n0 = sqrt(mu./A.^3);
24
25 tk = t - toa;
26
27 n = n0;
28
29 Mk = M0 + n.*tk;
30
31 Ek = m2e(Mk, e);
32
33 nuk = atan2( ( sqrt(1-e.^2).*sin(Ek)./(1-e.*cos(Ek))), ((cos(Ek)-e)./(1-e.*cos(Ek))) );
34
35 Phik = nuk + omega;
36
37 uk = Phik;
38
39 rk = A.*(1-e.*cos(Ek));
40
41 ik = i0 + di;
42
43 % positions in orbital plane
44 xks = rk.*cos(uk);
45 yks = rk.*sin(uk);
46
47
48 Omegak = Omega0 + (OmegaDot - OmegaEDot).*tk - OmegaEDot.*toa;
49
50
51 % satellite positions in earth fixed cs
52 xk = [xks.*cos(Omegak) - yks.*cos(ik).*sin(Omegak) ...
53         xks.*sin(Omegak) + yks.*cos(ik).*cos(Omegak) ...
54         yks.*sin(ik)];
```

posRecSat2el.m

```
1  % computes from position of receiver and position's of satellite
2  % correspondin elevation angles [rad]
3
4  function [el] = posRecSat2el(xR,xS)
5
6  %%%%%% compute elevation
7
8  % vector antenna - satellite
9  xRS = xS - ones(size(xS,1), 1)*xR';
10
11 % zenit distance
12 z = acos( dot(xRS', xR*ones(1,size(xRS,1))) ./...
13     (sqrt(dot(xRS',xRS')) .* sqrt(dot(xR*ones(1,size(xRS,1)),...
14     xR*ones(1,size(xRS,1))))) );
15 % elevation
16 el = (pi/2 - z)';
```

A.3 Extraction of signal noise

To extract the noise in the signal, two different strategies were discussed in this thesis. The data driven one is implemented in the `dataDriven.m` file, the model based extraction in the `modelBased.m` file. Both use the `polyfit.m` function, to fit a polynomial of n-th order into the data.

For the analysis the model based way, it was decided to the best way, so the code in the `iono.m` file, does calculate the ionospheric refraction for all satellites and frequencies recorded. After that, the input measEpoch data is extended by two columns, containing the information.

iono.m

```
clear all
close all
format long g
clc

%%% process geometry free combination

%%% additional columns
% 19 for code-measurement
% 20 for phase-measurement

%%% file definitions

disp('Start reading input files')

path = '../../00_Data/';
file = 'AAU_SEPT2340.10';

load(strcat(path, file, '__MeasEpoch1_processed.mat'));

% frequncy catalog
freqCatalog         = zeros(31,1);
freqCatalog(1:2)    = 1575.42;
freqCatalog(3:4)    = 1227.60;
freqCatalog(5)      = 1176.45;
freqCatalog(17:18)  = 1575.42;
freqCatalog(19:20)  = 1278.75;
freqCatalog(21)     = 1176.45;
freqCatalog(22)     = 1207.14;
freqCatalog(23)     = 1191.795;
freqCatalog = freqCatalog*10^6;

for PRN = 1:106

    posInMeas = find( measEpoch(:,4) == PRN );

    if( ¬isempty(posInMeas) )

        disp(['Processing satellite ', num2str(PRN)])

        sat = measEpoch ( posInMeas  ,: );

        % frequency 1
        posInMeasD1 = posInMeas( sat(:,7) == min(sat(:,7)) );
        data1 = sat ( sat(:,7) == min(sat(:,7)), : );
        % frewuency 2
        posInMeasD2 = posInMeas( sat(:,7) == max(sat(:,7)) );
        data2 = sat ( sat(:,7) == max(sat(:,7)), : );

        % get frequencies
        f1 = freqCatalog(data1(1,7)+1);
        f2 = freqCatalog(data2(1,7)+1);

        % geometry free combination
```

```
55          rhoG = data1(:,8) - data2(:,8);
56          phiG = data1(:,9) - data2(:,9);
57
58          % TEC
59          TEC = rhoG / ( 40.3 * ( 1/f1^2 - 1/f2^2 ) );
60
61          iotaF1 = 40.3 * TEC / f1^2;
62          iotaF2 = 40.3 * TEC / f2^2;
63
64          % write data back
65          measEpoch(posInMeasD1, 19) = -iotaF1;
66          measEpoch(posInMeasD2, 19) = -iotaF2;
67          measEpoch(posInMeasD1, 20) = iotaF1;
68          measEpoch(posInMeasD2, 20) = iotaF2;
69
70      end
71
72  end
73
74  disp('Saving output file')
75  save(strcat(path, file, '__MeasEpoch1_processed.mat'),'measEpoch')
```

dataDriven.m

```
1   clear all
2   close all
3   clc
4   format long g
5
6   %%%%%% data driven noise extraction via polynom
7
8   % %%% load measEpoch
9   path = '../../00_Data/';
10  file = 'LMU_STGT2000.10';
11
12  load(strcat(path, file, '__MeasEpoch_processed.mat'));
13
14
15  sat = measEpoch ( measEpoch(:,4) == 25 ,: );
16
17  data1 = sat ( sat(:,7) == 0  ,: );
18  data2 = sat ( sat(:,7) == 4 ,: );
19
20
21  %%% extract noise by fitting polynomial into data
22  % pick continous area
23
24  % data1 = data1(1:11000,:);
25  % data2 = data2(1:11000,:);
26
27  % data1 = data1(1:9000,:);
28  % data2 = data2(1:9000,:);
29
30  y = data1(1:end-1000,8);
31  x = data1(1:end-1000,2);
32
33  % degree = 3;
34
35  interval = 1000;
36
37  figure
38  plot(x,y)
39  % title('Original observation', 'Fontsize', 12)
40  ylabel('Pseudorange [m]', 'Fontsize', 12)
41  xlabel('Time Of Week [s]', 'Fontsize', 12)
42  print('-depsc', 'plots/dataDriven/orig')
43
```

```
44
45  %%% split data
46  measEpoch = y;
47
48  maxSize = 1200;
49  SizeMeasEpoch = size(measEpoch,1);
50
51  numParts = ceil(SizeMeasEpoch / maxSize);
52  measEpochNew = [];
53
54  for part=1:numParts
55      tic
56
57      % split data
58      if(part == 1)
59          data{part} = measEpoch(1+((part-1)*maxSize):part*maxSize-1 ,:);
60      elseif( part > 1 && part < numParts)
61          data{part} = measEpoch((part-1)*maxSize:part*maxSize-1 ,:);
62      elseif (part == numParts)
63          data{part} = measEpoch((part-1)*maxSize:end ,:);
64      end
65  end
66
67
68  for degree=1:4
69  xHat = [];
70  yHat = [];
71
72      for parts = 1:size(data,2)
73
74          xStep = (1:length(data{parts}))';
75
76          [xHatBuff, yHatBuff] = polyfit(xStep, data{parts}, degree);
77          xHat = [xHat; xHatBuff];
78          yHat = [yHat; yHatBuff];
79      end
80
81  noise = y - yHat;
82
83  sigma = std(noise);
84
85  figure
86  plot(x(1:end-1), noise(1:end-1))
87  title(['Residuales for polynomial of ', num2str(degree),...
88      '. order, GIOVE-B, code measurement'], 'Fontsize', 12)
89  ylabel('Pseudorange [m]', 'Fontsize', 12)
90  xlabel('Time Of Week [s]', 'Fontsize', 12)
91  text(min(x)+100,0, ['\sigma = ', num2str(sigma), ' m'],...
92      'BackgroundColor',[1 1 1], 'Fontsize', 13)
93  print('-depsc', ['plots/dataDriven/1_poly', num2str(degree)])
94  close all
95
96  end
```

modelBased.m

```
1  clear all
2  close all
3  clc
4  format long g
5
6  %%%%%% model based extraction
7
8  %%% load measEpoch
9  path = '../../00_Data/';
10 file = 'LMU_STGT2060.10';
11
```

```
12 load(strcat(path, file, '__MeasEpoch_processed.mat'));
13
14 %%% ChannelStatus column discription
15 % row
16 % 1 WNc[week]
17 % 2 TOW[s]
18 % 3 RxChannel
19 % 4 SVID
20 % 5 FreqNr
21 % 6 HealthStatus
22 % 7 Azimuth[deg]
23 % 8 Elevation[deg]
24 % 9 RiseSet
25 % 10 Antenna
26 % 11 TrackingStatus
27 % 12 PVTStatus
28 % 13 PVTInfo
29
30 sat = measEpoch ( measEpoch(:,4) == 1 ,: );
31 sat = sat(13675:end,:);
32 data1 = sat ( sat(:,7) == 0 ,: );
33 data2 = sat ( sat(:,7) == 4 ,: );
34
35
36 %%% filter signal type
37 % signal number row 7 in measEpoch
38 % 0  GPS_L1-CA   1575.42
39 % 1  GPS_L1-P(Y) 1575.42
40 % 2  GPS_L2-P(Y) 1227.60
41 % 3  GPS_L2C     1227.60
42 % 4  GPS_L5      1176.45
43 % 16 GAL_L1A     1575.42
44 % 17 GAL_L1BC    1575.42
45 % 20 GAL_E5a     1175.45
46 % 21 GAL_E5b     1207.14
47 % 22 GAL_E5      1191.795
48
49
50 c = 2.99792458e8;
51 f1 = 1575.42e6;
52 lambda1 = c / f1;
53 f2 = 1176.45e6;
54 lambda2 = c / f2;
55
56 % geometry free combination
57
58 rhoG = data1(:,8) - data2(:,8);
59 phiG = data1(:,9) - data2(:,9);
60
61
62 % ionosphere refraction
63
64 rhoIR = + f2^2/(f2^2-f1^2) * data1(:,8) - f2^2/(f2^2-f1^2) * data2(:,8);
65
66
67 % ionosphere free combination
68
69 rhoI = - f1^2/(f2^2-f1^2) * data1(:,8) + f2^2/(f2^2-f1^2) * data2(:,8);
70
71 % TEC
72
73 TECrho = rhoG / ( 40.3 * ( 1/f1^2 - 1/f2^2 ) );
74
75 iotaGroupf1 = 40.3 * TECrho / f1^2;
76
77
78 %%%
79 noiseRhof1 = data1(:,8) - rhoI - iotaGroupf1;
80 sigma = std(noiseRhof1);
```

```
81
82 [xHat, yHat] = polyfit(data1(:,2), iotaGroupf1, 2);
83
84
85
86 %%% plots
87
88 figure
89 plot(data1(:,2), iotaGroupf1)
90 % title('\iota^{code}')
91 xlabel('Time Of Week [s]', 'Fontsize', 12)
92 ylabel('Pseudorange [m]', 'Fontsize', 12)
93 % print('-depsc', 'plots/modelBased/iotaCode')
94
95 figure
96 hold on
97 plot(data1(:,2), iotaGroupf1)
98 plot(data1(:,2), yHat,'r','LineWidth',2)
99 xlabel('Time Of Week [s]', 'Fontsize', 12)
100 ylabel('Pseudorange [m]', 'Fontsize', 12)
101 legend('Ionospheric refraction','Polygon fit')
102 % print('-depsc', 'plots/modelBased/iotaCodeWithPoly')
103
104 figure
105 plot(data1(:,2), iotaGroupf1 - yHat)
106 xlabel('Time Of Week [s]', 'Fontsize', 12)
107 ylabel('Pseudorange [m]', 'Fontsize', 12)
108 % print('-depsc', 'plots/modelBased/noise')
```

A.3.1 Functions

polyfit.m

```
1  % fit polynom of degree n
2
3  function [xHat, yHat] = polyfit(x, y, degree)
4
5  % xmean = mean(x);
6  % x = x - xmean;
7
8  % estimate parameters
9  A = ones(length(x), 1);
10 for i=1:degree
11     A = [A x.^i];
12 end
13
14 xHat = (A'*A)\A'*y;
15
16 % x = x + xmean;
17
18 yHat = xHat(1);
19 for i=2:degree + 1
20     yHat = yHat + xHat(i)*x.^(i-1);
21 end
```

A.4 Analysis

In the `analysis.m` all calculation used for the analysis (Covariance function, Spectral density function, etc.) of the signal noise is included. To compare the analyzed signal with others, a exponential function is fitted into the data. This is done by the `expfit.m` function, which adjusts iteratively an exponential function and gives the estimated observations and the estimated coefficient of the exponential function back.

```
clear all
close all
clc
format long g


disp('Start reading input files')

path = '../../00_Data/';
file = 'LMU_STGT2060.10';

load(strcat(path, file, '__MeasEpoch_processed.mat'));
dt = 1; % sampling rate [s]

% signal number row 7 in measEpoch
% 0  GPS_L1-CA   1575.42
% 1  GPS_L1-P(Y) 1575.42
% 2  GPS_L2-P(Y) 1227.60
% 3  GPS_L2C     1227.60
% 4  GPS_L5      1176.45
% 16 GAL_L1A     1575.42
% 17 GAL_L1BC    1575.42
% 20 GAL_E5a     1175.45
% 21 GAL_E5b     1207.14
% 22 GAL_E5      1191.795

%%% choose satellite to analyze
PRN = 101;
FreqNr = 17;
sat = measEpoch(measEpoch(:,4) == PRN, :);
sat = sat(sat(:,7) == FreqNr, :);
measEpoch = sat(18234:end, :);

%%% choose measurement type
% 19 for code measurerment
% 20 for phase measuerement
iono = measEpoch(:, 19);

%%% computation of sigma for n parts of iono to create stochastic process


[xHat, yHat] = polyfit((1:size(iono,1))', iono, 2);

figure; hold on
plot(measEpoch(:,2), iono)
plot(measEpoch(:,2), yHat,'r','LineWidth',2)
% title('Ionospheric refraction')
xlabel('Time Of Week [s]', 'Fontsize', 12)
ylabel('[m]', 'Fontsize', 12)
legend('Ionospheric refraction','Polygon fit')
% print('-depsc', 'plots/ir')


iono = iono - yHat;

figure
plot(measEpoch(:,2), iono)
xlabel('Time Of Week [s]', 'Fontsize', 12)
ylabel('[m]', 'Fontsize', 12)
```

```
60 % title('Ionospheric noise reduced by geometry')
61 % print('-depsc', 'plots/iono')
62
63 p = [];
64
65 parts = 10;
66
67 numValues = floor(size(iono,1)/parts);
68
69 for i=1:parts
70
71     if(i==1)
72         data{i} = iono(1:numValues);
73     elseif(i>1 && i<parts)
74         data{i} = iono(numValues*(i-1)+1:numValues*i);
75     elseif(i==parts)
76         data{i} = iono(numValues*(i-1)+1:end);
77     end
78
79     sigma(i) = std(data{i});
80
81 end
82
83
84 sigma = sigma/max(sigma);
85
86 for i=1:size(data,2)
87     p = [p; data{i}/sigma(i)];
88 end
89
90
91 figure
92 hold on
93 plot(sigma)
94 % title('Normalized standard deviation for n parts')
95 % elevation
96 % plot( linspace(1, parts, size(measEpoch,1))  , measEpoch(:,18)/max(measEpoch(:,18)), 'r')
97 xlabel('# of applied parts', 'Fontsize', 12)
98 ylabel('Normalisation factor', 'Fontsize', 12)
99 % print('-depsc', 'plots/norm')
100
101
102 figure
103 plot(measEpoch(:,2), p)
104 % title('Stationary Process')
105 xlabel('Time Of Week [s]', 'Fontsize', 12)
106 ylabel('[m]', 'Fontsize', 12)
107 % print('-depsc', 'plots/statprocess')
108
109
110 %%% statistics
111
112 data = p;
113 m = mean(data);
114
115
116 %%% empirical coveriance function
117 N = size(data,1);
118
119
120
121 C = zeros(size(data,1),1);
122
123 for n=0:N-1
124
125     for i=1:N-n
126
127         C(n+1) = C(n+1) + 1/(N-n) * (data(i)-m) * (data(i+n)-m);
128
```

```
    end

end


figure
plot((0:size(C,1)-1)*dt, C)
% title('empirical covariance function')
xlim([-length(C)*0.02 length(C)])
xlabel('n', 'Fontsize', 12)
ylabel('C(n) [m^2]', 'Fontsize', 12)
% print('-depsc', 'plots/cov')

%%% empirical power spectral density function

N = 500; % samples

S = fftshift(fft(data,N));

S = abs(S/N);

SHat = S.^2;

% units
nu = pi/dt;
domega = nu/(N);
omega = [-N/2:N/2-1] * domega;
f = omega/(2*pi);

figure
plot(f,SHat)
% plot([-N/2:N/2-1]/N,SHat)
% title('empirical power spectral density function')
xlabel('[Hz]', 'Fontsize', 12)
ylabel('[m^2/Hz]', 'Fontsize', 12)
xlim([-0.15 0.15])
% print('-depsc', 'plots/pdf')


%%% fit exponential function

y = C(1:5000);

x = (1:size(y,1))' * dt;

% initial values
% xIni = [1 10 .001]';

xIni = [0 y(1)]';
xIni(3) = sum( log( abs(y) / xIni(2)) ./ [1:(length(y))]' ) / size(y,1);


[xHat, yHat] = expfit(x, y, xIni, 0.95);

xHat

% correlation length
posCorr = find( min( abs( yHat - max(yHat) / 2 ) ) ==...
                abs( yHat - max(yHat) / 2 ) );

corrLength = posCorr*dt

%plot
figure
hold on
plot(x, y)
plot(x, yHat, 'r', 'LineWidth',2)
stem(posCorr, yHat(posCorr), 'g--','LineWidth',1, 'MarkerFaceColor','green')
legend('Covariance function','Esitmated Markow process','Correlation length')
```

```
xlim([-length(x)*0.02 length(x)])
xlabel('n', 'Fontsize', 12)
ylabel('C(n) [m^2]', 'Fontsize', 12)
% title('Exponential function')
% print('-depsc', 'plots/exp')


%%% transform exponetial function to power spectral density function
Se = fftshift(fft(yHat,N));
Se = abs(Se/N);

figure; hold on
plot(f, SHat)
plot(f, Se,'r', 'LineWidth',2)
% title('empirical power spectral density function')
xlabel('[Hz]', 'Fontsize', 12)
ylabel('[m^2/Hz]', 'Fontsize', 12)
xlim([-0.15 0.15])
% print('-depsc', 'plots/pdf2')
```

A.4.1 Functions

expfit.m

```
% Fit of an exponential function
% [xHat, yHat] = expfit(x, y, xIni)
% Model y = a0 + a1 * exp( a2*x )
% Input:    x
%           y
%           xIni = [a0 a1 a2]'; initial values
%           alpha, for significance test < 1
% Output:   xHat: adjusted parameters
%           yHat: adjusted observations

function [xHat, yHat] = expfit(x, y, xIni, alpha)

xHat = xIni;
diff = 1;

while diff > 10^-10

    % mask because coefficient a0 is only applied on first observation
    mask = [1; zeros(size(y,1)-1, 1)];

    dy = y - (xHat(1).*mask + xHat(2) * exp( xHat(3)*x) );

    A = [mask exp(xHat(3)*x) xHat(2)*x.*exp(xHat(3)*x) ];

    dxHat = (A'*A)\A'*dy;

    diff = dxHat'*dxHat;

    xHat = xHat + dxHat;

end
A = [mask exp(xHat(3)*x) xHat(2)*x.*exp(xHat(3)*x) ];

% coefficient a0 will not be taken into account for yHat
yHat = xHat(2) * exp( xHat(3)*x );

% significance test
disp(['Siginificance test with alpha = ', num2str(alpha*100), '%.']);

f = size(y,1) - 3;

t = tinv(alpha, f);
v = yHat - y;
```

```
44
45 s0 = sqrt(v'*v/f);
46 sxHat = sqrt( s0^2 * diag(inv(A'*A)) );
47 ti = abs(xHat./sxHat);
48
49 res = ti < t;
50
51 for i=1:length(res)
52     if( res(i) == 1 )
53         disp(['Parameter ',num2str(i), ' is not significant.']);
54     else
55         disp(['Parameter ',num2str(i), ' is significant.']);
56     end
57 end
58 propability = tcdf(ti, f);
```