Software Receiver

By Group 947

Aalborg University
Institute of Electronic Systems

Title :
EGNOS Software Receiver

Project Period :
September 1st 2004 - January 4th 2005

Project Group :
Kostas Dragūnas
Vitalijus Linikas
Xiangfu Meng
Usman Ahmed Zahidi
Jose Luis Corral Sanchez

Supervisors :
Laust Olsen
Kjeld Hermansen

Number of reports printed: 9

Number of pages in report: 112

Number of pages in appendix: 40

Total number of pages: 177

Abstract

Recent developments and modifications in satellite-based positioning systems and new augmentation systems raise the need for a flexible and easily modifiable technology to cope up with these amendments. The software receiver is an enabling technology for flexible implementation of a receiver in which significant amount of the signal processing is accomplished in software rather than in hardware. This report documents a prototype software EGNOS receiver (EGNOS receiver stands for EGNOS-capable GPS receiver). The CDMA signal structure associated with GPS and EGNOS provides demanding computational requirements, therefore, the receiver design is considered in post process mode. Receiver is able to do GPS and EGNOS signal acquisition and tracking. Real GPS and EGNOS data obtained from an RF front end is used for simulation of this algorithms. Satellites ranging capabilities are not considered. Obtained EGNOS signal tracking performance is not satisfactory enough to extract real EGNOS data from the signal and requires improvements. Simulation of EGNOS corrections application to GPS pseudoranges is done by using real EGNOS data recorded from hardware EGNOS receiver. Raw GPS measurements obtained from the same hardware receiver is used for this simulation as well. This allowed testing and development of algorithms with actual GPS and EGNOS data to verify their performance. Results show considerable improvement in the GPS positioning accuracy. Future refinement will include the transition from a prototype to a complete GPS EGNOS-enabled software receiver. The application has been designed in order to show advantages of software receivers, where new algorithms and signals can be easily added or previously modified.

Preface

This report presents the work of group 947 of the GPS specialization at the Faculty of Engineering and Science, Institute of Electronic Systems, Department of Communication, Aalborg University, Denmark.

The report contains 8 chapters and 5 appendices. Chapter 1 introduces the general project matter. Chapter 2 covers the basic characteristics of GPS and EGNOS systems and describes the software receiver approach perspectives. Chapter 3 gives the project problem formulation. Chapter 4 covers analysis. Chapter 5 deals with design and implementation. Chapter 6 represents the tests and relevant results. Chapter 7 considers future perspectives of this project. Chapter 8 concludes the project work. Finally, additional information related with the project issues is provided in appendices. A complete reference list where additional details can be found if desired is provided as well.

The project was proposed by Kai Borre.

January 4, 2005

Kostas Dragūnas

Xiangfu Meng

Vitalijus Linikas

Usman Ahmed Zahidi

Jose Luis Corral Sanchez

Acknowledgements

Project group acknowledges the assistance of Peter Rinder and Nikolaj Bertelsen, research assistants at Danish GPS Center at Aalborg University, Kai Borre, Head of Danish GPS Center, for proposing the project, and our supervisors Laust and Kjeld for their guidance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of new satellite-based navigation systems, modernization plans in existing ones, such as Global Positioning System (GPS) and new augmentation systems, such as European Geostationary Navigation Overlay Service (EGNOS) have opened a new era in the research of satellite-based global positioning and navigation systems. Entirely new navigation systems, augmentation systems and newly introduced frequency bands require the researchers to experiment the core signal processing algorithms, executing in the receivers, in order to gain flexibility. Conventional hardware receivers are built on Application Specific Integrated Circuits (ASIC) and therefore new features can not be introduced easily. The development of fast digital processing units such as microprocessors, Digital Signal Processing (DSP) chips and Field Programmable Gate Arrays (FPGAs) motivated the researchers to work on the development of software GPS receivers.

A software receiver comprises of a hardware unit usually termed as, Radio Frequency (RF) front end, which amplifies the signal received by the antenna, down-converts its carrier frequency to an acceptable Intermediate Frequency (IF) and performs the digitization. After the digitization at acceptable sampling rate, typically a microprocessor is used for satellite signal acquisition, tracking, demodulation and position calculation on the basis of the data in the navigation message.

Aiding to GPS software receiver can come in two ways: in terms of reduced processing time, where less time would be required to solve for the user position, and/or in terms of accuracy, where the user's GPS position could be improved.

The primary advantage of a software receiver is its flexibility. However, the computing time is still being a major concern in achieving good real-time performance. Keeping the increase in cheaper and faster microprocessor development it is assessed that this drawback will be overcame in near future. The software receiver may also be implemented on a fast DSP chip or FPGA, the performance of these is better than microprocessors.

Accuracy has been a major concern in GPS positioning for a long time. Several error correction methodologies have been proposed that includes differencing by various means. Differenc-

ing has been a successful technique for error corrections and therefore was used on larger scale accounts for the development of Ground-Based Augmentation Systems (GBAS) and Satellite-Based Augmentation Systems (SBAS). SBAS can improve GPS accuracy and extend it by other means.

Several attempts have been made to implement the GPS software receivers, but least are known to have the feature of SBAS corrections in software receivers. The aim of this project is to implement the SBAS corrections in a software receiver by processing the SBAS geostationary satellites signals and data and applying ionospheric, tropospheric and other corrections provided by SBAS signals to GPS pseudoranges. Specifically, we have focussed on the implementation of EGNOS in this project.

The EGNOS signal has three main advantages:

1. Compliant with international standards and interoperable with similar systems.
2. Design is based on GPS.
3. Provides enhanced accuracy and integrity.

Benefits of EGNOS are expected in a lot of applications. EGNOS target markets encompass aviation, maritime, road and rail transport, personal and consumer applications. As EGNOS has been designed principally as a safety critical system [2], the target market segments for EGNOS reside mainly in the safety-of-life and performance critical applications.

Some examples of possible EGNOS applications are provided below:

- **Aeronautics**

  The increasing air traffic requires a very precise positioning system guiding the airplanes accurately throughout their whole flights, especially the takeoff and landing operations. EGNOS will enable the plane to take the optimum route and therefore cutting down the operation costs and reducing air pollution.

  EGNOS is designed to assist navigation both en-route as well as during landing. The potential benefits will assist air traffic control to cope with increased traffic as well as improving safety and reducing the infrastructure needed on the ground.

- **Land transport**

  EGNOS is one of the keys to managing land transport whether it be by road, rail or inland waterways. It will increase both the capacity and the safety of land transport. Not only airlines but also companies which operate transport services need to know where their vehicles are at all times, as do other public services such as the police and the ambulance and taxi services.

As well as improving safety, EGNOS will be an invaluable aid to managing transport operations. Managers will be able to know exactly when a consignment has been held up and its exact location. This will also improve customer services as clients can be notified of delays and the reason for them and when necessary breakdown crews can be sent out immediately.

- **Time standard**

  Computer and telecommunication networks around the world need an extremely accurate clock reference, a kind of "world speaking clock". EGNOS will be able to broadcast a reliable time standard with unprecedented accuracy.

- **Miscellaneous**

  EGNOS has many other potential uses. It can help farmers in aerial crop spraying, fishermen to locate their catch and the police to detect fraud. EGNOS can also be used for leisure activities such as hiking, sailing and climbing. Every day more and more potential uses are being found for EGNOS.

  Software receiver approach will make EGNOS applications cheaper and thus more popular.

Summary

Recent changes in the existing satellite-based positioning and navigation systems and their augmentation systems requires more flexible receivers development technology. Software approach is the one. However, its flexibility is a trade-off between processing time and accuracy. The SBAS systems such as EGNOS provide differential signal corrections with the aid of geostationary satellites. SBAS helps in improving the accuracy in GPS positioning. This project emphasizes on the implementation of EGNOS Software Receiver. EGNOS will find a lot of applications. Software receiver approach will make EGNOS applications more popular.

Chapter 2

Background

In this chapter we give the background theory relevant to the development of EGNOS software receiver. We describe the Global Navigation Satellite System (GNSS), an integration of existing satellite systems, the precision of Global Positioning System and the sources of errors that give rise to the development of augmentation systems. We also shortly describe the theory of signal acquisition, tracking and demodulated data processing. Readers interested in getting a deeper understanding of the underlying theory may refer to the literature mentioned in this chapter.

2.1 Literature Study

Software approach concept is already around for about 10 years. There has been much work done regarding methods and techniques in the general area of this project by various researchers. However, books dealing with the project issues are quite few. There are books about GPS, but they deal with either specific parts of GPS or other aspects of GPS like navigation, geodesy, or other.

Many research groups have undertaken the development of software receivers in recent years (e.g. Akos, Ledvina, Jovancevic). It is not possible to mention all of them. Recently research in this area started at Aalborg university as well. Papers published by researchers are good information sources.

There are some fundamental books dealing with principles of GPS. The two-volume set "Global Positioning System: Theory and Applications", edited by Parkinson, Spilker, Axelrad and Enge, published in 1996 by American Institute of Aeronautics and Astronautics (AIAA), will remain an authoritative work for years to come. Other good resources are "Understanding GPS Principles and Applications" edited by Elliott D.Kaplan and "Fundamentals of Global Positioning System Receivers. A Software approach" by James Bao and Yen Tsui, where design of software receiver in post process mode is described.

GPS Interface Control Document (ICD) is useful when dealing with GPS concepts. However, main interest of the project was EGNOS signal processing, therefore mostly "Radio Technical

Commission for Aeronautics (RTCA): DO-229C" document was studied, which deals with Minimum Operational Performance Standards (MOPS) for SBAS systems. Also it is relevant to mention ION GPS publications, which were very useful.

All above mentioned references and many others were used throughout the project. Reference list is provided in the end of the report.

2.2 Feasibility Study

Compared to conventional hardware GPS receivers, software receivers are more suitable for research and development of new signal processing algorithms [3]. A conventional hardware receiver is normally based on ASIC chips of which the RF part comprises the down conversion, signal filtering, Automatic Gain Control (AGC), sampling, and quantization, while the signal processing part comprises all of the baseband signal processing and navigation solution computation. Although ASIC chips now have very high processing speeds, the chip design is fixed, and a chip cannot be modified to experiment with new signal tracking techniques. Pure software GPS receivers, on the contrary, realize all GPS functions in software, which obviously makes the above experimentation possible. Even more important is that, besides a front end, a software receiver does not require any other hardware, so it can work just as a software module and share system resources such as power, onboard memory and processor, with other subsystems.

Hardware approach limits the flexibility of the GPS receiver architecture for particular applications of interest. Software approach is very flexible. If a different processing architecture would be desired, it would simply be a matter of a software change rather than replacing physical components. This will also provide much more rapid prototyping since new signals/algorithms only need to be implemented in software. In addition, simulation and verification of algorithms are tied closer together in the development cycle, since the actual software used for prototyping can be reused for production of the system.

One of the difficulties associated with a software receiver implementation is the required programmable processing power, particularly for wide-bandwidth spread-spectrum systems such as GPS and SBAS. While software radios can offer tremendous flexibility in the signal processing, their throughput will also be less than that from a component-specific design or a specific architecture. The trade-off between flexibility and available processing power is illustrated in Figure 2.1.

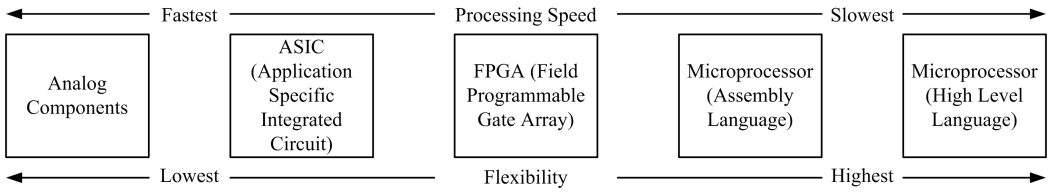


Figure 2.1. Processing speed vs. flexibility using hardware components or software tools

As microprocessors continue to evolve, software designs become more feasible and attractive.

2.3 Global Navigation Satellite System (GNSS)

There are two GNSS systems known today. These are GLONASS and GPS. Russian system GLONASS is not fully operational due to incomplete constellation of satellites. New European Galileo system is in developing stage, and it is planned to be completed in 2008. The only fully operational positioning system today is the American GPS.

GPS is a satellite-based system that provides positioning and timing information to users worldwide. System comprises of three segments: space, ground and user segment. User segment states for a GPS receiver, that receives signals from satellites and based on the measurements made and information extracted from these signals computes an user position.

GPS is a passive system in which users only need to receive and process the transmission to utilize the system. No users transmission or response is required for operation. Originally developed for the United States military, the system now provides a military specific component known as GPS Precise Positioning Service (GPS-PPS) and a civilian component known as GPS Standard Positioning Service (GPS-SPS). The focus of this report is on the publicly available GPS-SPS which is also known as Coarse Acquisition (C/A) code GPS.

Although the GPS is the most accurate worldwide navigation system yet developed, it can still exhibit significant errors.

2.4 Error Sources of GPS

There are 6 main error sources in GPS [4]:

1. **Ephemeris data.** Errors in the transmitted location of the satellite.

   Ephemeris is so called Keplerian elements for satellite position prediction. It is valid for a few hours and used to compute satellite positions with prediction into the future. Unfortunately, the prediction is not very precise and is tend to degrade. Thus, the satellite position errors will appear and will grow slowly with time from the last upload from the ground control station to the next upload.

2. **Satellite clock.** Errors in the transmitted clock.

   GPS is based on a very precise clocks which let users to compute the range from satellite to receiver. Here the atomic clocks are used to measure time with very high precision. However, these clocks are not perfect and any slight offset from the true GPS network time will cause inaccuracies in user position determination.

3. **Ionosphere.** Errors in GPS observables caused by ionospheric effects.

   Ionosphere is the layer of atmosphere which starts approximately at 50 km above the Earth and extends to 1000 km or more [5]. The layer contains free electrons which can be ionized

depending on the Sun and its activity. This effect will have an influence on the propagation of radio frequency electromagnetic waves used in GPS signal transmission. Passing this layer signals will delay and thus, the time for signal to reach user will increase. This will lead to an error in user position.

4. **Troposphere.** Errors in the GPS observables caused by tropospheric effects.

   Another delay will occur when signal passes the troposphere. This layer is in the lower part of the atmosphere and it has variations in temperature, pressure, and humidity which all contribute to variations in the speed of radio waves.

5. **Multipath.** Errors caused by reflected signals entering the receiver antenna.

   Multipath is the error caused by reflected signals entering the front end of the receiver and masking the real signal. It is more or less the local phenomenon and corrections of these error are not supported by SBAS service.

6. **Receiver.** Errors in the receiver's measurements.

   Receiver errors is also a local phenomenon which differs from receiver to receiver. These errors can be caused by thermal noise, software accuracy, and inter-channel biases in the receiver.

The standard errors in GPS are shown in the Table 2.1 [4].

Error source	**1σ error[1], m**
Ephemeris data	2.1
Satellite clock	2.1
Ionosphere	4.0
Troposphere	0.7
Multipath	1.4
Receiver measurements	0.5
UERE, RMS[2]	5.3
Vertical 1σ errors	12.8
Horizontal 1σ errors	10.2

Table 2.1. Standard error model-L1

For more information on error sources of GPS please refer to [4].

[1] A one sigma (1σ) probability has an associated value of 68.3% which essentially means that approximately 68% of the measured positions will have the desired position tolerance

[2] UERE - the User Equivalent Range Error; RMS - Root Mean Square

2.5 Satellite Based Augmentation System (SBAS)

There are three known SBAS developed or are under development. These include Wide Area Augmentation System (WAAS) developed by USA, EGNOS developed by Europe and Multi-functional Transport Satellite Based Augmentation System (MSAS) developed by Japan. These systems mainly cover areas of their design, that is WAAS covers North America, EGNOS Europe and MSAS - Japan (Figure 2.2).

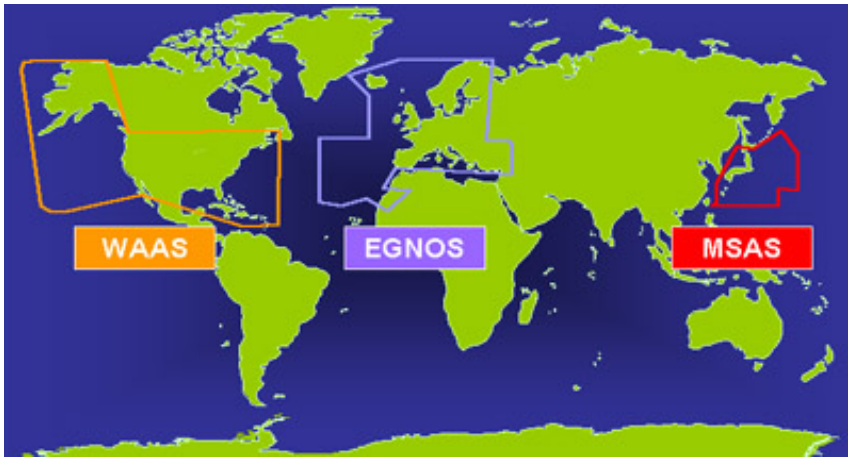


Figure 2.2. SBASs coverage areas

SBAS provides free-to-air differential correction service. The system augments GPS with additional signals that increase the reliability, integrity, precision and availability of civil GPS signals. SBAS services are common in design and their performance. Report considers new European version of SBAS service EGNOS. A lot of information is available on U.S WAAS system, however practically not many references can be found on EGNOS. Fortunately SBAS services are common and only slight differences exist. SBAS signal and service in this report will be referred to as EGNOS or just SBAS, although mostly all relevant information is obtained from references for WAAS system.

EGNOS Signal in Space (SIS) is broadcast from geo-stationary satellites, equipped with navigation payloads to broadcast a GPS look-alike signal containing integrity and wide-area differential corrections [1]. The operational system uses three satellites to disseminate this data: Inmarsat III Atlantic Ocean Region-East (AOR-E) at 15.5° W, ESA ARTEMIS at 21.5°E and Inmarsat III F5 (IOR-W) at 25.0°E. Moreover, the EGNOS System Test Bed (ESTB) signal is still available and can be obtained through Inmarsat III F1 (IOR-E) satellite at 64.01°E (Figure 2.3).

EGNOS signals will be available in areas where other Differential GPS (DGPS) sources are not available. GPS receivers that are able to track EGNOS signals will provide a free and integrated real-time position solution without the need for additional equipment.

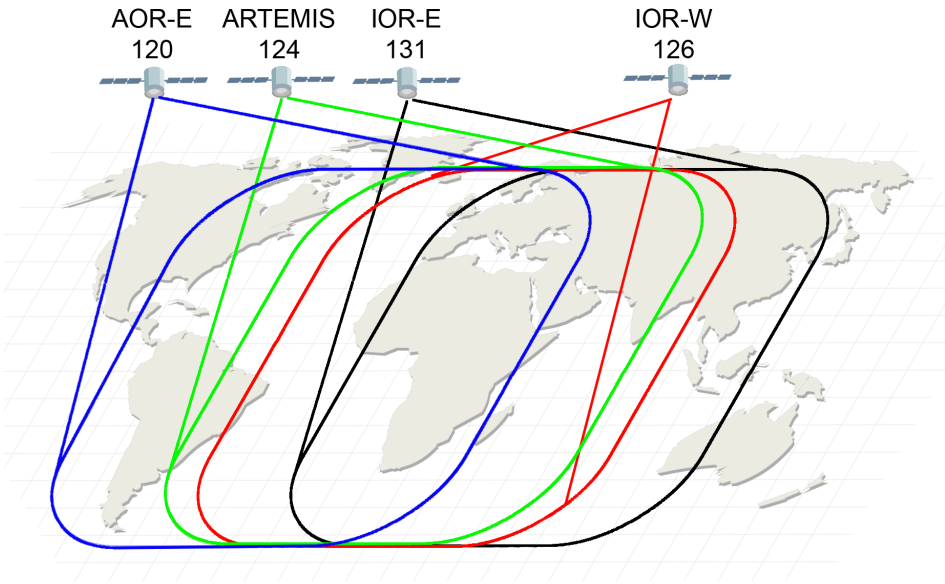


Figure 2.3. EGNOS satellites and coverage

The EGNOS network consists of 34 ground Reference and Integrity Monitoring Stations (RIMS), 4 Mission Control Centers (MCC), 6 Navigation Land Earth Stations (NLES) and other facilities (ASQF and PACF) located in Europe (Figure 2.4). The network of RIMS stations all over its coverage area receives GPS signals from all GPS satellites in view. Then, the GPS data is sent to MCC stations, where GPS differential corrections and GPS satellite health data is computed and transmitted to NLES stations, which will send this data to the geo-stationary satellites. These satellites broadcast the information to all EGNOS-capable GPS receivers, which then decode EGNOS signal to obtain differential corrections.

The actual accuracy of EGNOS corrected positions also depends on the GPS receiver used and its location relative to the EGNOS satellites.

EGNOS signals are broadcast free to all EGNOS-enabled receivers.

Full initial operational capability of the system has not been declared yet (refer to Appendix C), but this restriction applies only to some safety-of-life uses of EGNOS. EGNOS signals can be tracked in areas outside the Europe, but the accuracy and reliability of the signals may be significantly reduced.

If a GPS user wants to benefit from a source of free differential corrections, it is necessary to upgrade GPS receiver to support EGNOS corrections.

EGNOS receivers have the SBAS option enabled as a part of their standard configuration. EGNOS receiver will be able to track WAAS signal, as well as WAAS-enabled receiver will be able to track EGNOS. MOPS of the SBAS services are the same and should be applied when

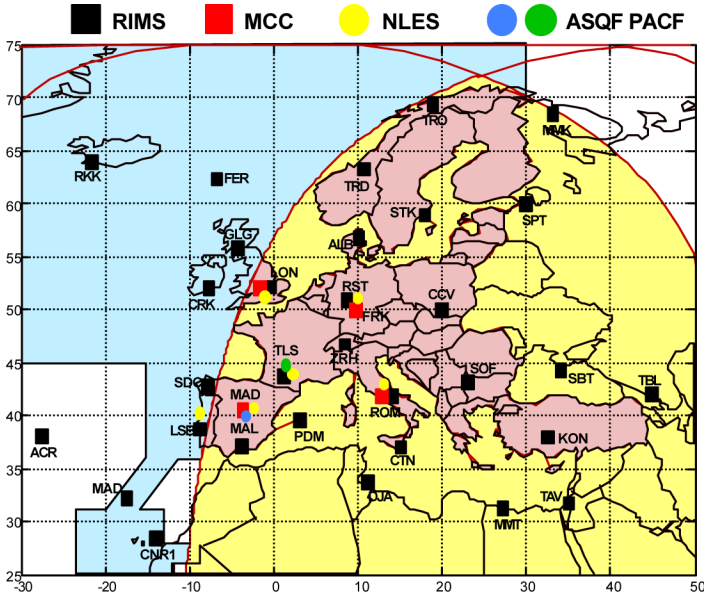


Figure 2.4. EGNOS structure [1]

designing an SBAS-capable receiver [6]. To get the precision and integrity benefits from the EGNOS service, at least one channel in the GPS receiver must be dedicated to EGNOS satellite to decode the EGNOS message. The remaining channels can select GPS satellites that provide the best geometry for position calculations.

2.6 Software Receiver

2.6.1 Software radio

The Software Defined Radio (SDR) is an enabling technology that is being applied across a wide range of application domains and provides comparatively efficient, inexpensive and flexible solutions to several systems. It enables to have electronic equipments that can be dynamically programmed in software, enhancing the capability of adapting and to provide new features and services with minimal efforts. Its basic concept is built upon two basic principles: move the Analog-to-Digital converter (ADC) as close to the antenna as possible and process the resulting samples. The implementation of a software GPS Receiver (SGR) is ideal for embedding in a Software Defined Radio. The general structure of the software receiver is shown in Figure 2.5.

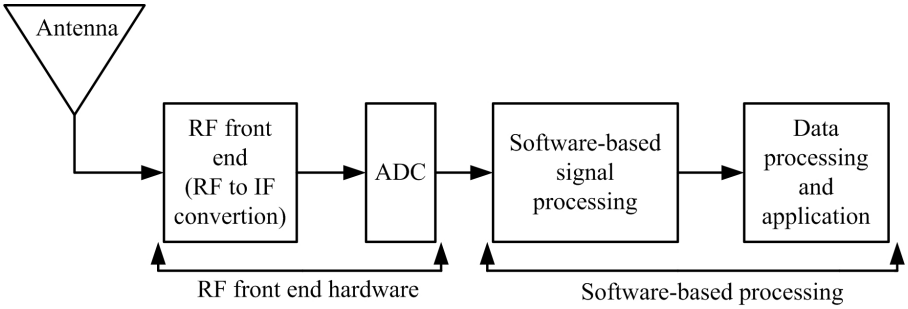


Figure 2.5. General architecture of software-based receiver

2.6.2 GPS/EGNOS software receiver

EGNOS-enabled GPS receiver is generally called just an EGNOS receiver. This project deals with the implementation of a prototype software EGNOS receiver. In order to understand the receiver development procedures, which are described later in this report, the GPS and EGNOS systems characteristics are shortly covered below.

All GPS satellites are broadcasting synchronized ranging codes and navigation data on two frequencies, L1 at 1575.42 MHz and L2 at 1227.6 MHz.

The signaling is Code Division Multiple Access (CDMA), which enables multiple signal transmission in the same frequency channel with an acceptable interference. The multiple access capability is important for GPS because a user may receive multiple signals simultaneously from different satellites, wherein all signals occupy the same frequency channel and are continuous.

All GPS satellites are broadcast on the same carrier frequencies and have a unique code which is referred to as ranging code. The ranging codes are known by all receivers and are replicated internally allowing demodulation of the data from specific satellites.

The navigation data message, among other items, contains information required to compute the satellite position in space (ephemeris data) and predicted satellite clock correction terms [7]. The content of the navigation message is continuously updated by the GPS control segment, and broadcast to the users by the GPS satellites. The whole GPS navigation message is transmitted in 12.5 minutes. It consists of 25 subframes (5 frames consisting of 5 subframes each). Each subframe is broadcast every 6 seconds. In order to compute position at least 3 subframes are required to be received by the receiver. That makes at least 30 seconds of data, since it is unknown which subframe will be received first.

By calculating the timing difference in message arrival and departure time, the pseudorange, that is the approximate distance between the receiver and the satellite, can be measured, which is the basis for the position calculation. The basic principle for obtaining the pseudorange is so-called "code-correlation" technique whereby the incoming code from the satellite is correlated with a replica of the corresponding code generated inside the receiver, as depicted in Figure 2.6.

Both codes are generated using the same mathematical algorithm. The time shift (dt)

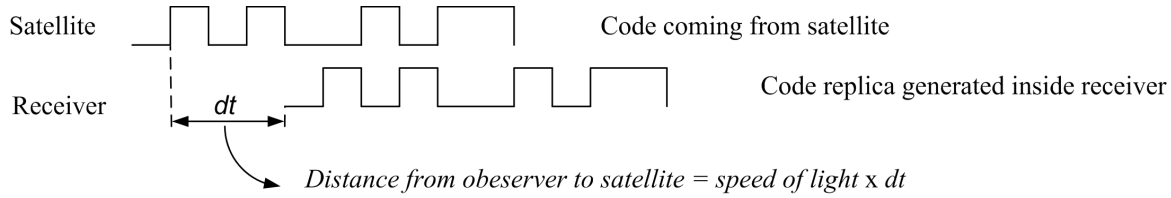


Figure 2.6. Received signal alignment

required to align the two codes is, in principle, the time required by the signal carrying the code to travel from the satellite to the receiver. Multiplying *dt* with the speed of light results in an estimate of the range. This range is referred to as a pseudorange because it is still biased by the time offset (or mis-synchronization) between the satellite clock and receiver clock used to measure the time delay. At least four pseudorange measurements from four different satellites are necessary in order to obtain receiver position. The synchronization error is determined by the receiver along with its position coordinates from the pseudorange measurements. When the receiver clock bias is determined, the "true range" to all received satellites can be specified. However, this "true range" is still biased by several other effects including ionospheric and tropospheric delay, multipath, and receiver noise (refer to section 2.4).

Most civilian GPS receivers intended for navigation applications only observe the C/A-code pseudorange, and hence are referred to as single-frequency navigation receivers. The C/A code is 1 ms long and its chipping[3]rate is 1.023 MHz. This project does not deal with measurement of pseudoranges, however this would be interesting for future refinement.

The civil GPS service provides a horizontal accuracy of 10 meters and timing accuracy at the nanosecond level. However, as described in previous sections, the GPS is not an error free system. This is where EGNOS is helpful. With the aid of EGNOS the civil GPS performance can be improved significantly.

There are two main types of correction data provided by EGNOS, namely fast corrections and long-term corrections. The fast corrections contain errors from sources which are frequently changing like satellite clock errors. The long-term corrections contain errors from sources which are changing slowly like long-term satellite clock drift and ephemeris errors. More details about EGNOS corrections can be found in chapter 4.

The signals broadcasted by EGNOS geostationary satellites is designed such that any standard GPS receiver with minimum hardware modifications will be able to track the EGNOS signals. Therefore, GPS frequency, type of modulation and ranging codes are retained in EGNOS signals. However the data format is different when compared to GPS as it has to accommodate additional information and integrity data.

The EGNOS broadcasts a signal with carrier frequency of 1575.42 MHz (GPS L1). The modulation technique for the transmission of code and data, used in EGNOS signals is similar

[3]Chip is an equivalent of a bit

to the GPS, although data transmission rate is different. Message symbols at a rate of 500 symbols-per-second (sps), 50 bits-per-second (bps) in the case of GPS, are added modulo-2 to a 1023-bit PRN code, which then is Bi-Phase Shift-Keyed (BPSK) onto the carrier at a rate of 1.023 Mega-chips per second. EGNOS transmits messages every second. Three messages make one frame.

The general structure of a GPS EGNOS-enabled software receiver is illustrated below (Figure 2.7).

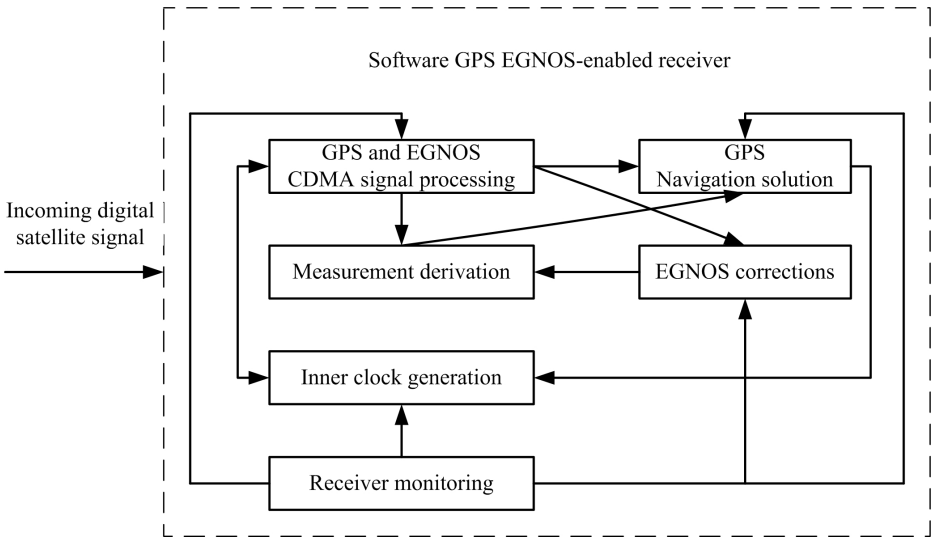


Figure 2.7. General software-based GPS EGNOS-enabled receiver block diagram

Only a digitized IF signal can be processed by a software receiver. Therefore prior to entering software-based processing module the GPS and EGNOS L1 signal is frequency down-converted by RF front-end, sampled and quantized. Further signal is processed by using standard CDMA signal processing techniques. CDMA signal processing algorithms are described in the next section. Like GPS, EGNOS is a CDMA system. Such receiver provides EGNOS corrected pseudoranges, from which more accurate GPS position is computed. An inner clock based on the incoming IF signal samples is generated as the time reference in such receivers because no physical clock is available in the software receiver as in hardware receivers. Also receivers usually have some monitoring mechanism in order to switch between different states in the receiver.

2.6.3 GPS/EGNOS signal processing

The signal processing of a CDMA system can be functionally divided into three parts which are signal acquisition, code/carrier tracking, data demodulation and data processing. Subsequent sections give an insight to these functions.

Acquisition

The navigation message is a low frequency signal while the C/A code and carrier are high frequency signals. To receive a GPS or EGNOS signal, these two high frequency components have to be acquired and tracked.

As previously described in the case of GPS and EGNOS each satellite is broadcasting a unique ranging code. This code is called a Pseudorandom Noise (PRN) code, since it has a noise like properties. The PRN codes have the property that the correlation between any pair of codes is very low and the cross-correlation[4] is high. This is is why all satellites can share the same carrier frequencies. Satellites in view must be determined by correlating the received signal with the local replicas.

Signal acquisition is a two-dimensional search process in which a replica code and carrier are aligned with the digitized IF signal [8]. The basic acquisition scheme is illustrated in Figure 2.8.

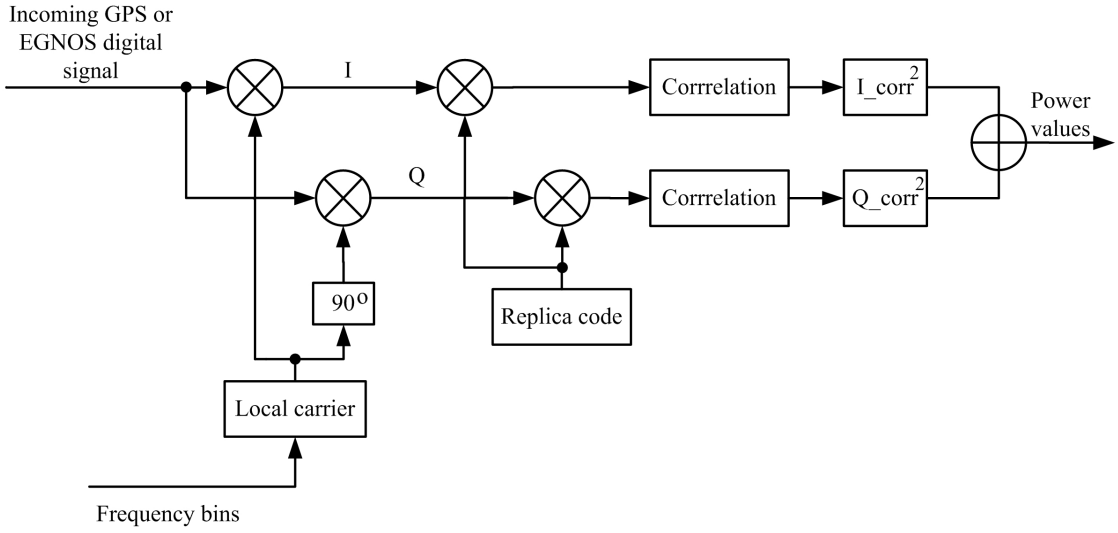


Figure 2.8. Basic acquisition scheme

First the incoming signal is converted to baseband, that is multiplied with complex exponential local carrier. An In-phase (I) and Quadrature-phase (Q) components are obtained. Further the signal is correlated with local code replica. The obtained signals samples are then squared to reduce the noise. The correct signal alignment and its presence in the incoming data is identified by measurement of the output power of the correlators.

Determining the proper alignment of the received code provides the code phase. The other parameter of interest is the exact carrier frequency. Although this is typically known for a CDMA system, a search is still required as line-of-sight dynamics can introduce an unknown

[4]Correlation between same codes

frequency Doppler shift. The acquisition search is conducted in predefined frequency bins and usually through all code phases (refer to chapter 4) in order to find the satellite signal. When both the code phase and carrier Doppler match the incoming signal, the signal is despread.

Once the acquisition of a signal has been achieved, the processing can proceed to a tracking mode in which two loops operate in parallel. The loops adjust the locally generated signal for proper alignment with the incoming satellite signal.

Signal tracking and data demodulation

The acquisition approach discussed earlier gives the initial estimates of the carrier Doppler and C/A code offset. Then control is handled over to tracking module to track variations of carrier frequency (or phase) and code offset due to line-of-sight movement between satellites and the receiver.

The receiver uses its tracking module to make pseudorange measurement and to extract the broadcast message. This is done through the use of tracking loops [7]. A tracking loop is a mechanism which permits a receiver to "tune into" or track a signal which is changing in frequency or in time. It is a feedback device which basically compares an incoming (external) signal against a locally-produced (internal) signal, generates an error signal which is the difference between the two, and uses this signal to adjust the internal signal to match the external one in such a way that the error is reduced to zero or minimized. Ordinary receivers usually contain two kinds of tracking loops: the code tracking loop and the carrier tracking loop.

The code tracking loop is used to align PRN code sequence that is present in the signal coming from satellite with an identical one which is generated within the receiver. Alignment is achieved by appropriately shifting the receiver-generated code chips in time so that a particular chip in the sequence is generated at the same time instant its twin arrives from the satellite. An error signal is generated by code loop discriminator and the internally generated code is adjusted to keep alignment with the external one (Figure 2.9). In this way the replicated code sequence is locked to the sequence in the incoming signal.

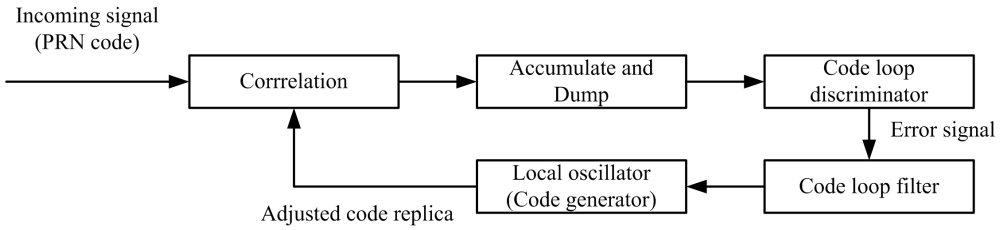


Figure 2.9. Generic code tracking loop

The signals from other satellites will have essentially no effect on the tracking process because the PRN codes of all the satellites were chosen to be orthogonal to each other. This orthogonality property means that a very low output is always produced by the correlator whenever the code

sequences used by two different satellites are compared.

Since the chips in the satellite code sequences are generated at precisely known instants of time, the alignment of the receiver and satellite code sequences also gives a reading of satellite clock at time of signal generation.

Once the code tracking loop is locked, the PRN code can be removed from the signal by correlating it with the locally generated one and filtering the resultant signal. Correlation usually just stands for multiplication of signals. It is also known as mixing. This procedure as mentioned earlier despreads the signal. The despreaded signal then passes to the carrier tracking loop which demodulates or extracts the satellite message by adjusting the phase or the frequency of the receiver's local oscillator signal with the incoming signal. If the phase or frequency of the oscillator signal is not correct , this is detected by the discriminator in the carrier tracking loop and a correction signal is then applied to the oscillator (Figure 2.10).

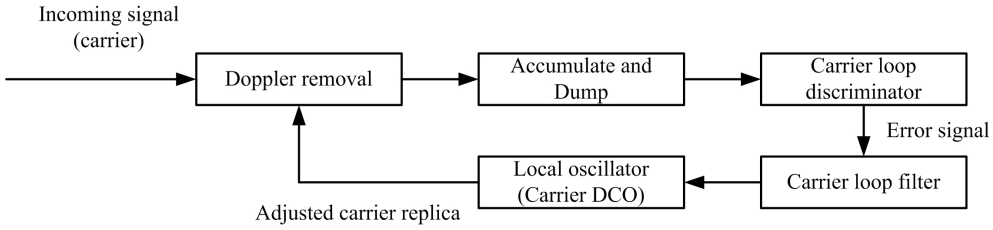


Figure 2.10. Generic carrier tracking loop

Once the oscillator is locked to the satellite signal, it will continue to follow the variations in the carrier as the range to the satellite changes.

In software receivers the replica carrier is synthesized by the Digitally Controlled Oscillator[5] (DCO) and discrete sine and cosine mapping functions and the replica codes are synthesized by the code generator.

Conventional Delay Lock Loop (DLL), to track C/A code, and Frequency Lock Loop (FLL) or Phase Lock Loop (PLL), to track carrier, can be implemented in the software EGNOS receiver.

Most implementations of carrier tracking use the Costas loop, a variation of the PLL designed for binary-modulated signals such as those transmitted by the GPS and the EGNOS satellites.

As described above to track an incoming signal, both carrier and C/A code need to be matched by the locally generated counterparts. As a result, the carrier locked loop (FLL or PLL) and the code lock loop (DLL) need to be coupled together [9] (Figure 2.11).

When the incoming signal is multiplied with the local counterpart carrier, carrier from the external signal is removed if both signals matches. Then what is left is pure PRN code (and data bits). This is the input to code tracking loop. When the incoming signal is multiplied with the local code counterpart, PRN code from the signal is removed if both signals matches. Then what is left is carrier. It is an input to carrier tracking loop. The base band I and Q signals are

[5]Hardware receivers employ Voltage Control Oscillators (VSOs) or Number Control Oscillators (NCOs)

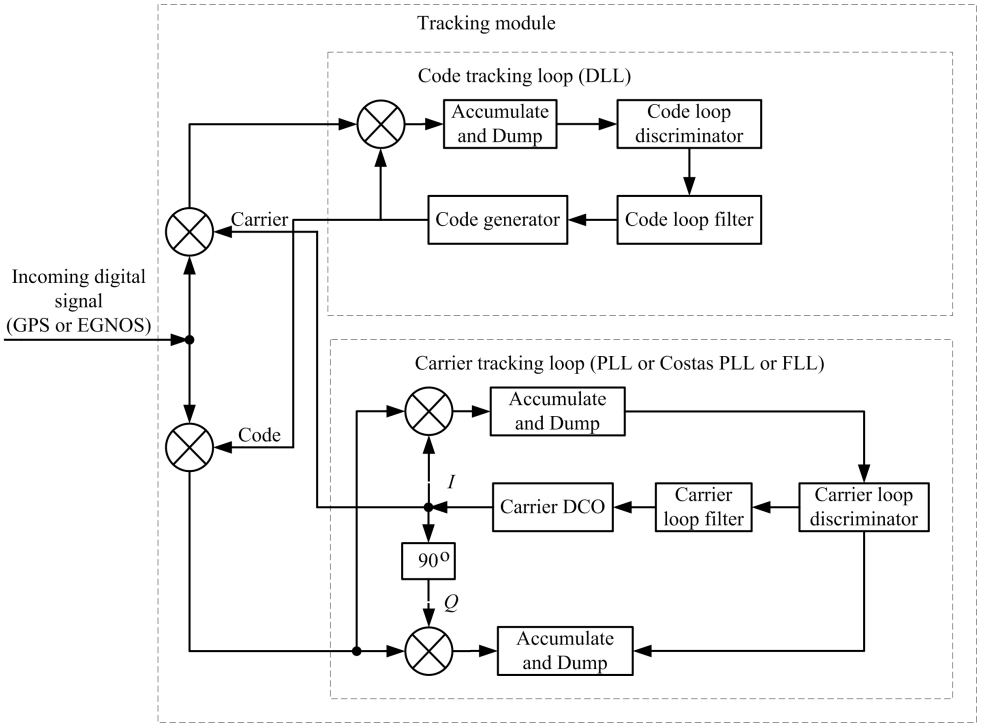


Figure 2.11. Receiver's tracking module

accumulated and dumped. The code generator usually produces three different codes. These are early, prompt and late. For simplicity this is not shown on the diagram. Accumulated and dumped signal is applied to loop discriminators, by which the error signals are generated. Further the error signal are filtered and the carrier and code are adjusted to coincide with the incoming signal. The discriminators provide the noisy estimates of the errors between the input signal and locally generated replicas. The classification of discriminators is defined by the type of tracking loop. A PLL discriminator produces a phase error and a FLL discriminator produces a frequency error. The loop filters perform filtering on the signal errors after the discriminators since the discriminators outputs are very noisy. The objective of the loop filter is to reduce the noise in order to produce an accurate estimate of the original signal and its output. The filtered error estimations are then input to the carrier DCO and code generator to adjust the locally generated carrier and code phase to match the input signal.

Once one of the loops loses the lock, the other one will lose the lock as well. Generally, the carrier loop is a weaker loop because the carrier wavelength is much shorter than the code chip length and the carrier loop needs to track all dynamics while the code loop needs only to track the dynamics difference between carrier loop and code loop when carrier aiding is applied to code loop.

As soon as the carrier tracking loop locks onto a satellite the signal is demodulated and the data bits in the broadcast message are subsequently extracted using standard techniques of bit synchronization (refer to section 4.4).

Demodulated data processing

The project mainly deals with EGNOS data processing and application. More details about data processing can be found in chapter 4. Here we just want to emphasize the significant difference between GPS and EGNOS systems.

After EGNOS signal is demodulated it has to be additionally decoded, and only then broadcast data can be extracted and processed. That is not the case with GPS, where the broadcast data can be extracted immediately after signal demodulation.

EGNOS data is convolutionally encoded. The convolution codes are often used in digital transmission systems and new GNSS and SBAS systems make use of them.

The operation of a general convolutional encoder as shown in Figure 2.12 can be described as follows [10]. The input to the encoder is a sequence m of message digits and the output is the corresponding sequence c of code digits. At any time unit, a block of k message digits (called a message block) is fed into encoder, and a block of n code digits (called a code block) is generated at the output of the encoder, where $k < n$. The n-digit code block depends not only on the k-digit message block of the same time unit, but also on the previous $(N-1)$ message blocks. The code generated by the above encoder is called an (n, k) convolutional code of constraint length N blocks (or nN digits). The rate of this code is $R = k/n$.

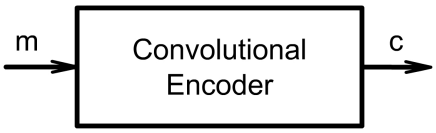


Figure 2.12. A general convolutional encoder

Since the encoding of a convolutional code is performed block by block (k message digits to n code digits at a time), the received sequence is therefore decoded a block of n digits at a time. The encoded string consists of blocks of length n each one representing k information symbols. The process of decoding a convolutional code is conveniently described as a search for a chain that disagrees with the received sequence in the smallest number of positions.

The convolutional encoder achieves error free transmission by adding enough redundancy to the source symbols. The choice of the convolutional code is application dependent and varies with the frequency characteristics of the transmission medium as well as the desired transmission rate.

The Viterbi's algorithm is a method commonly used for decoding bit streams encoded by convolutional coders. For codes with short constraint lengths, Viterbi's decoding method is very effective. The details of a particular decoder algorithm depend on the encoder.

2.7 Summary

Software GPS receivers are more suitable for research and development. Implementation of GPS receiver in software is feasible. GPS receiver can be made EGNOS-capable. GPS and EGNOS systems make use of standard CDMA signal processing techniques: 1) acquisition 2) PRN code and carrier tracking, and data demodulation, and 3) data processing. EGNOS data is convolutionally encoded. Viterbi's algorithm is applied to decode the received EGNOS signal.

Chapter 3

Problem Formulation

This chapter contains the problem description and problem statement delimitation outlining the scopes of the project, the limitations and segregating development stages.

3.1 Problem Description

The GPS observables used in computing position (the ranges to each satellites being tracked) using the software GPS receiver could be adjusted for common mode errors, such as ionospheric, tropospheric delays, satellite clock biases by applying EGNOS corrections. Error corrections are computed by the EGNOS monitoring stations, observing the GPS satellites at a known fixed locations, and uploaded to EGNOS geostationary satellites. EGNOS corrections can be received from geosationary satellites by any GPS EGNOS-enabled receiver, processed and applied to GPS observables. This approach will improve the resulting GPS position solution. A stand-alone software GPS receiver can be then designed which will be able to provide an accuracy comparable to hardware GPS receivers. The accuracy of software GPS receivers is not completely as such of hardware receivers available on the market today. Therefore, EGNOS capability will improve the performance of a software receiver.

Some GPS receivers, both hardware and software, provide sufficient accuracy and actually do not require any aiding. In this case SBAS augmentation can be used for nonprecision approach, where stand-alone GPS receiver provides sufficient accuracy, and SBAS data only needs to provide integrity.

Precision approach, where the SBAS must provide vector corrections to achieve the accuracy requirements is involved when stand-alone GPS receiver is not able to provide sufficient accuracy [4].

In both ways the software GPS receiver will gain from SBAS augmentation. Software receiver which provides sufficient positioning accuracy itself will gain in terms of integrity, otherwise the receiver will gain in terms of accuracy improvement.

This project mainly addresses civil GPS accuracy improvement.

3.2 Problem Statement

The objective of this project is to enhance the precision of a civil software GPS receiver by applying the EGNOS corrections.

3.3 Problem Delimitation

The concept of the software receiver covers a very broad area, and so the scope for development in this area is also large. The title of our project is "EGNOS Software Receiver", however, it is not realistic to implement fully functional receiver in 4 month period. Based on these considerations the scope has been limited to design a prototype EGNOS software receiver. A prototype EGNOS receiver is developed as a software application, running on a personal computer architecture. The CDMA signal structure associated with GPS and EGNOS provides demanding computational requirements, therefore, the receiver design is considered in post process mode.

Main focus is put on the processing of the signal broadcast by EGNOS geostationary satellites, however the algorithms for GPS signals are also implemented, as there has been plenty of literature and algorithm designs available on GPS software receivers.

Final goal of the project is a design and simulation of a prototype receiver that runs in a postprocess mode and is able to do GPS and EGNOS signals acquisition and tracking. Simulation of EGNOS corrections application to GPS observables is done with the help of data recorded from any other receiver, which is able to provide necessary measurements.

The project main issue is the software implementation for signal processing. Signal processing is done using high level programming language, that is MATLAB. A survey of the tasks that the software shall implement are presented in the following section.

3.4 Technical Specifications

In order to achieve the defined objectives the task can be segregated as below:

Signal acquisition: A coarse synchronization process giving estimates for PRN code offset and the carrier Doppler has to be implemented. Carrier frequency estimate has to be refined before switching to tracking mode.

Signal tracking: Variations in the carrier Doppler and code offset due to line-of-sight dynamics between the satellite and the receiver have to be tracked. Another important function of the tracking loops is to demodulate the data from the incoming signal.

Bit and frame synchronization: Databit timing is subject to data bit offset ambiguity. Therefore, it is necessary to determine databit timing and to eliminate the databit offset ambiguity.

EGNOS data decoding: Viterbi's decoding scheme has to be applied to demodulated EGNOS data before trying to extract any data bits from demodulated signal.

Application of data: Simulation of EGNOS corrections application to raw GPS measurements have to be implemented. All data for simulation is obtained from a hardware or a software fully functional EGNOS receiver.

3.5 Development Stages

Due to its complexity the project was subdivided into two main development stages:

1. Develop a prototype GPS EGNOS-enabled receiver.
2. Simulate pseudaorange corrections obtained from a hardware or a software GPS EGNOS-enabled receiver. Analyze GPS position accuracy improvements.

3.6 Summary

Civil software GPS receivers require accuracy improvements. EGNOS provided error corrections can solve this problem. The CDMA signal structure associated with GPS and EGNOS signals processing gives a very heavy computational burden, thus, the receiver design is considered in post process mode. The implementation is done in MATLAB. Final result of the work is GPS EGNOS-enabled prototype receiver which is able to acquire and track EGNOS and GPS satellites. Simulation of EGNOS corrections to GPS pseudoranges is done on recorded data obtained from a fully functional EGNOS receiver.

Chapter 4

Analysis

In this chapter we present the analysis of the project goal. Decisions concerning design and implementation will be made. Analysis such as, general problem analysis, signal acquisition, signal tracking, EGNOS data decoding, EGNOS data processing and application will be presented. Chapter describes acquisition and tracking algorithms, bit synchronization techniques. Description about how EGNOS system works by analyzing WAAS MOPS [6] and other related sources have been mentioned.

4.1 Problem Analysis

Software implementation of GPS receiver gives great flexibility, it enables to test and combine different algorithms. Recently many promising algorithms are developed. It is possible to process signal in time or frequency domain, or to perform acquisition in time and tracking in frequency, or acquisition in frequency and tracking in time. Their general analysis is covered, only.

The receiver can not process the signal directly at frequency of its transmission (refer 2.6.2), thus the RF front-end device down converts the signal from 1.5GHz L1 frequency to a much lower Intermediate Frequency of 3.563MHz. During this conversion process, the signal is also digitized (A/D conversion) at 14bits rate and sampled at 11.999MHz frequency. The down-converted and digitized signal is used for further processing.

Actually the fewer the number of quantization bits, the easier the digital process becomes. However, at the same time the fewer the number of quantization bits, the greater the signal degradation caused by quantization. Normally one-bit and two-bit quantizations are used in many commercial receivers [3]. Higher sampling frequency also gives heavier computational load, at the same time allowing more precise signal processing.

The already recorded data was available for the purposes of the first project development stage (as in chapter 3). This project does not deal with the hardware part of the software receiver and therefore, any data parameters. The data was accepted like it is (Table 4.1).

It is necessary to emphasize that in order to control the performance of the algorithms it is

IF frequency	3.563MHz
Sampling frequency	11.999MHz
Number of quantization bits	14

Table 4.1. Simulation data parameters for the first stage of the project

necessary to get acquainted with the hardware. Possibility of changing hardware parameters, such as sampling frequency and quantization bit rate would give an extreme advantage for the development. Unfortunately, project group had no such possibility. On the other hand availability of recorded data allowed completely concentrate on the software algorithms implementation and do not spend time on any other matters.

First the short analysis is provided on the staff required to consider before starting the software receiver development. Later, different possibilities of CDMA signal processing are elaborated and finally, the simulation of EGNOS corrections application is analyzed in more details.

SBAS system requirements

The GPS/SBAS receivers should be compatible with the GPS/SBAS MOPS [6]. An EGNOS receiver that is SBAS MOPS compatible will track WAAS signals. According to [6] MOPS compatible SBAS receiver should be able to acquire and track all PRN codes that are assigned to SBAS services. Receivers should be designed to acquire and track all of the defined SBAS codes, not only the allocated ones, since the future SBAS satellites could use defined codes. The following PRN numbers are allocated for SBAS: from 120 to 138. Not all of them are currently in use. PRNs assigned to GPS are: from 1 to 37. However, the GPS system makes use only of the first 32. Other signals are maintained for future system upgrades. When the signal comes from the satellite it is completely unknown which PRN number it has. Therefore, all 32 GPS and 19 SBAS satellites are required to be considered in processing if a receiver design is done according to SBAS MOPS. However, as was described in project problem delimitation, our main interest is EGNOS signal processing. As it is mentioned in chapter 2, currently EGNOS system makes use of three geostationary satellites. Since the prototype receiver design is planned, the decision was made to consider processing only of these three EGNOS satellites and not of all SBAS satellites, taking in mind that some of them are not used recently at all and will be employed only in future. A software implementation of receiver gives flexibility as additional signal to processing modules can be easily added and if a new EGNOS satellite appears, algorithms can be easily modified to process this new signal. Hardware receivers do not have this option.

It is necessary to notice that EGNOS main purpose is not just accuracy improvement, but rather integrity data. Simple exclusion of unhealthy GPS satellite, flagged by EGNOS system, from position computation improves the resulting position accuracy.

EGNOS signal acquisition and tracking, mostly is common to GPS, since the EGNOS system is GPS based. EGNOS signal also provides ranging capabilities, as GPS does, however, in this project this issue is not covered.

In order to track GPS receiver to track EGNOS signal in GPS receiver it has to dedicate one channel for EGNOS satellite. The receiver runs in post-process mode. In post process mode number of channels is not limited.

Notice that EGNOS only provides the correction terms, thus enhancing potential positioning accuracy. The GPS receiver still has to receive the GPS navigation message and solve the position autonomously.

PRN codes

The EGNOS codes are identified in three ways [6]:

1. PRN number.
2. G2 delay in chips.
3. Initial G2 state.

The EGNOS signals use the similar C/A codes used in the GPS. They belong to the family of Gold codes. G2 stands for shift register used in PRN codes generators and characterizes code generation alternatives.

The following PRNs are used by EGNOS: 120,124,126 (Figure 2.3). The PRN 131 belongs to ESTB satellite (chapter 2) and actually is not anymore part of the EGNOS system. ESTB satellite broadcasts will be stopped soon [2].

GPS civil codes mainly are generated by using so-called two tap selection generators. The EGNOS codes cannot be implemented by using these. The EGNOS C/A code generator can be implemented by using a programmable G2 shift register delay with a single output, or a programmable G2 initial shift register state with a single output. The GPS codes can be generated with either of these implementations as well [6].

For implementation of code generator the programmable G2 delay was chosen (Figure 4.1).

Table 4.2 specifies the EGNOS codes and their G2 shift register delay, called code delay.

EGNOS codes	G2 delay (chips)
120	145
124	237
126	886

Table 4.2. EGNOS C/A codes and octal G2 delay

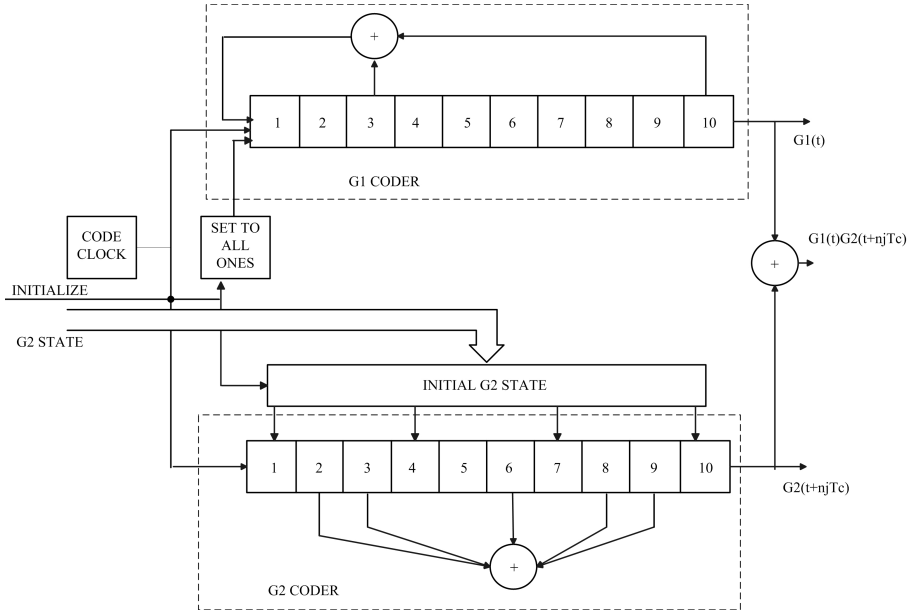


Figure 4.1. A programmable initial G2 state coder

Main EGNOS signal characteristic

EGNOS uses geostationary satellites and therefore, satellite overall Doppler shift reduces to 210 Hz at L1 in the worst case.

EGNOS signal is designed in a way that it do not interfere with GPS signal [4]. A typical GPS antenna will receive EGNOS signal.

The received power level through a typical GPS antenna will vary from -161 dbW to -157 dBW depending on elevation angle. The signal is assumed to be received at low elevation angle of 5 deg.

The EGNOS power levels are generally weaker than the current and expected power levels for GPS. Even the highest possible SBAS power level is no more than 7 dB above the minimum specified GPS C/A power level of -164 dBW.

The correlation level of the weakest GPS C/A signal will always be at least 17dB over the strongest SBAS signal. Therefore, EGNOS received power levels are generally below that of GPS.

4.2 Acquisition

The two-dimensional search space covering the full range of the uncertainty of code and carrier Doppler frequency needs to be defined before acquisition. It is difficult to determine the code

offset because this offset is a function of the starting point and changes at the chipping rate with time [8]. Thus, the code search space typically includes all possible code offset values. The change of Doppler is a function of user dynamics. Therefore, the frequency search space can be reduced if an initial estimate of Doppler is known in advance. However, usually it is unknown. The Doppler search space is usually from -10kHz to +10kHz [11]. The frequency resolution is determined by the coherent integration time (or dwell time) [9]. The relationship is roughly given as:

$$\Delta f = \frac{2}{3T} \tag{4.1}$$

where Δf is the frequency bin width in Hz and T is the predetection integration time in seconds. More roughly the frequency bin is given simply as:

$$\Delta f = \frac{1}{2T} \tag{4.2}$$

If $T = 1ms$, then one frequency bin is approximately given as 500 Hz. With the Doppler search space defined above this will lead to a total of 41 frequency bins. There is a tradeoff between the predetection time integration and acquisition speed. Longer integration can provide better frequency resolution and higher sensitivity, but it needs to search a greater number of bins and requires more time.

Various acquisition methods with search and detect strategies have been proposed in the literature. In a conventional receiver, acquisition is performed using a carrier replica and a C/A-code replica. It resides physically on each bin for a predetermined dwell time and, thus, the acquisition time is the product of the dwell time and the number of bins. Consequently, the acquisition is very long. In a post-mission software receiver, the samples of the incoming signal are first read into a buffer, making block processing possible. A popular block signal acquisition technique is the Discrete Fourier Transform (DFT), which is able to search all possible code offsets in one DFT-based computation, dramatically reducing the computational load [11]. Usually Fast Fourier Transform (FFT) is used instead of DFT.

The local data is a combination of the carrier and C/A-code. The circular convolution gives the acquisition results of all possible code offsets at a specific carrier frequency.

This method can give the correlation value at all possible codes in one value. Thus a FFT-based acquisition is chosen.

The correlation level of the EGNOS signal is weaker than that of GPS. The way to acquire weak signal is to accumulate energy from consecutive time slots. Usually acquisition methods are limited by data bit transition in the signal. Some accumulation methods can run beyond the data bit transition boundaries, by taking the magnitudes. Another accumulation may need more precise demodulating frequency. The general expression of signal energy accumulation in frequency domain is given by:

$$|R[m]| = \sum_{j=0}^{K-1} |\{IFFT[FFT((\sum_{l_j=0}^{N-1} s_{l_j}[n]) \cdot \cos[\Omega_j n] + i \cdot (\sum_{l_j=0}^{N-1} s_{l_j}[n]) \cdot \sin[\Omega_j n]) \cdot FFT^*(CA[n])]\}| \tag{4.3}$$

where

R	-	frequency correlator output
FFT	-	Fast Fourier Transform
$IFFT$	-	Inverse Fourier Transform
s_{l_j}	-	the incoming satellite signal
CA	-	C/A code
N	-	number of coherent integrations
K	-	number of noncoherent integrations
$*$	-	stands for complex conjugate

The EGNOS signal energy needs to be accumulated over a period of time because it is too weak for direct acquisition.

In the case of normal acquisition $N = 1$, which approximately corresponds to the signal bandwidth of 1 ms. When $N = 10$, it shrinks into 1/10 kHz. This narrow bandwidth is actually beneficial to the weak signal acquisition, although it also requires thorough frequency search. Block size N makes it difficult for FFT block to process on. Block addition technique can be used in this case, that is, 10 ms input signal is summed to one ms block size. The whole data block is overlapped and added to fit the FFT blocks. However, the frequency search step for 10 ms coherent data block integration according to equation 4.2 is 50Hz. That corresponds to 401 frequency search bins, that still gives heavy computation. Although frequency domain process can speed up the acquisition, large amount of computation is still unaffordable. Therefore, it is reasonable to set $K = 10$ as the accumulation time window size. The block size N is set to 1, which means that a block is only a single millisecond of signal in this case. Advantage of taking $N = 1$ with large K is that data bits would not cancel the accumulation energy since the magnitudes instead of complex correlation functions are summed up. Therefore, this accumulation can go on as long as the receiver hardware can support. In this case accumulation is after the correlation calculation, therefore the method can be called Post Correlation Overlapping.

For this method the equation 4.3 reduces to:

$$|R[m]| = \sum_{j=0}^{9} |\{IFFT[FFT((s_j[n]) \cdot \cos[\Omega_j n] + i \cdot s_j[n] \cdot \sin[\Omega_j n]) \cdot FFT^*(CA[n])]\}| \tag{4.4}$$

The frequency domain noncoherent correlator is shown below (Figure 4.2)

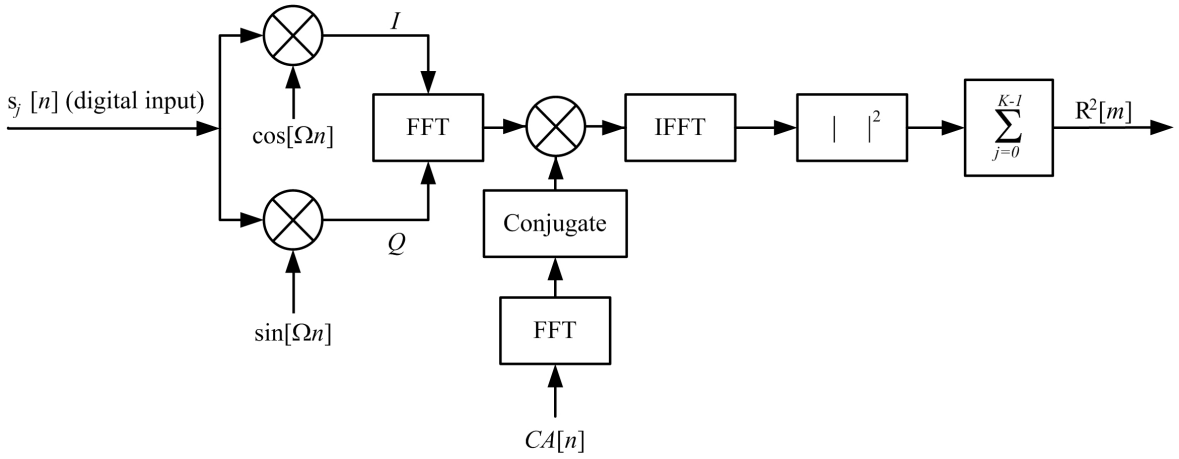


Figure 4.2. Noncoherent correlator in frequency domain

In general, the acquisition program has less sensitivity than the tracking program. From experimental results it appears that as long as the acquisition program can find a signal, the tracking program can track it without any difficulty [11].

In acquisition, search is done in an open loop manner according to pre-defined directions and steps, which are typically large. The search process will be determined whenever a signal is acquired. Such an operation can be sustained only if the signal is of certain strength.

All PRN codes are known and are generated or stored in GPS satellite signal receivers carried by ground observers [12]. Real-time generation of PRN codes in software receiver is an expensive process. It is much cheaper to generate all the codes off-line and to store them in memory. Therefore this approach is chosen.

The original C/A code chip rate is 1023 chips per millisecond but the received signal is up sampled to be 11999 points per millisecond so the receiver has to upsample the local code as well. In this case one ms of data block makes 11999 points. Direct frequency domain acquisition of the C/A code needs 11999 point FFT processing block. The FFT computation of such number of points is not efficient. FFT operations are fast when applied to radix 2 data sequences. 11999 is evidently not the one of them. A new technique is to do the frequency domain correlation with much cheaper FFT blocks.

To optimize acquisition process it is possible to use Modified Averaging Correlation method to perform block level signal C/A code acquisition in frequency domain with smaller FFT blocks. The method retains the signal to noise ratio observed in more expensive correlators. With this method frequency domain signal processing will become more feasible and popular.

The Averaging Correlation method was proposed in [13]. The Modified Averaging Correlation method is described in [14].

The acquisition process has to decide on the presence or absence of the signal. Thus the decision logic is required to be implemented for acquisition process (Figure 4.3). For this purpose it is necessary to compute acquisition Thresholds (TH) before acquisition, and to compute acquisition Test Statistics (TS) during the acquisition process.

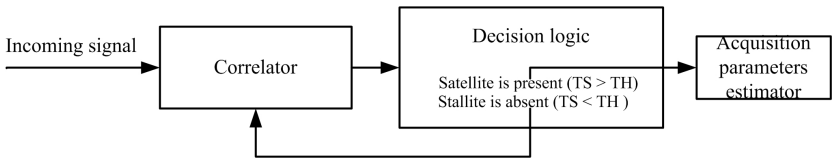


Figure 4.3. Signal acquisition process

If a TS is higher than the TH the satellite is declared to be present, if TS is lower than TH the satellite is declared to be absent in the incoming signal.

For the software acquisition correlation functions the traditional definition of Signal-to-Noise (SNR) ratio is not usually used. Instead, the strength ratio of peak to second largest peak or the average of all remaining peaks is introduced to determine the signal strength over the noise.

If the satellite signal is found, then its parameters are estimated and process switches to tracking loops.

4.3 Tracking

4.3.1 Open loop tracking

In contrast to acquisition, tracking search is usually done in a closed-loop manner where each step and direction are calculated based on the current and previous search results and the resulting search step is typically small. However, the easiest way to perform signal tracking is to repeat acquisition on each and every 1 ms segment of GPS data and then follow the correlation peak from segment to segment [15]. If there is no filtering/average (i.e., memory) from one ms to the next, it implements an open loop tracking. If this method is used in frequency domain then there is no need to divide acquisition and tracking into two different. When the possible range for code phase and Doppler frequency is covered, the capture of the GPS signal is guaranteed, thus enabling it to operate under high dynamics. However, when the GPS signal is weak, the peak follower may erroneously chase noise. Code chip boundary crossing introduces a modulation onto the correlation peak value. Discrete finite search steps in time and frequency introduce loss of SNR, together with frequency bin jumps and the associated phase jumps.

The open loop tracking is easy to implement but it uses too many search steps in both the time and frequency domains. This may be necessary to produce the full delay-Doppler map for the initial acquisition but it is totally wasteful once the signal is detected. Furthermore, the discrete search steps introduce large code delay and Doppler frequency errors, which are treated for search speed in acquisition (thus tolerated) but they are too coarse in tracking, resulting in

unnecessary loss of SNR and errors in signal parameter estimation. In addition, this open loop tracking implementation has no ability to coast through temporarily loss of signal and may have to declare no signal or pick up a wrong peak. To solve these and other problems, the closed-loop tracking implementation is presented.

4.3.2 Closed loop tracking

Conventional closed loop tracking algorithms PLL and DLL are widely known and approved since they are used by hardware receivers for many years. The choice was made to implement these in a software receiver.

The type of a tracking loop is defined by the discriminator and filter used in its design. The order of the loop is defined by the loop filter's order [9]. The programmable designs of the carrier predetection integrators, the carrier loop discriminators, and the carrier loop filters characterize the receiver carrier tracking loop. The carrier loop is always the weak link in a stand-alone GPS receiver.

The carrier loop discriminator defines the type of tracking loop as a PLL, a Costas PLL (which is a PLL type discriminator that tolerates the presence of data modulation on the base-band signal), or a FLL (refer chapter 2).

The PLL and the Costas loops are accurate but are more sensitive to dynamic stress than the FLL. The PLL and Costas loop discriminators produce phase errors as their outputs. The FLL discriminator produces a frequency error. Because of this, there is also a difference in the architecture of the loop filter.

There is a paradox that the GPS receiver designer must solve in the design of the predetection integration, discriminator, and loop filter functions of the carrier tracking loops. To tolerate dynamic stress, the predetection integration time should be short, the discriminator should be a FLL, and the carrier loop filter bandwidth should be wide. However, for the carrier Doppler phase measurements to be accurate (have low noise), the predetection integration time should be long, the discriminator should be a PLL, and the carrier loop filter noise bandwidth should be narrow. In practice some compromise must be made to resolve this paradox. A weel-designed receiver will close its carrier tracking loops with short predetection integration times, using a FLL and a wide band carrier loop filter. Then it will systematically transition into a Costas PLL with its predetection bandwidth and carrier tracking loop bandwidth set as narrow as the anticipated dynamics permits. It will also have a provision to revert to FLL operation during high-dynamics stress periods.

Normally, only Costas carrier tracking loops used in GPS receivers because the 50-Hz navigation message data modulation signal remains after the carrier and code signals have been wiped off the incoming satellite signals. Costas loops are insensitive to 180-deg phase reversals in the I and Q signals if the predetection integration times of the I and Q signals do not straddle the data bit transitions. Costas PLLs are sensitive to dynamic stress. This project implementation deals with a stationary receiver, so it was decided to implement only a PLL. Second order Costas PLL was chosen.

For the code tracking loop implementation ordinary DLL, with certain modifications in respect to software design, was chosen.

In contrast to acquisition, tracking of the signal is considered in time domain. Time domain correlators are easy to implement, since only addition and multiplication operations are needed.

In the tracking loops the signal is demodulated. Tracked signal is 1000 Hz long and requires to be converted to a data message length. That is 50 Hz in the case of GPS and 500 Hz in the case of EGNOS. For this bits synchronization is necessary.

4.4 Bit Synchronization

Bit synchronization algorithms include the Histogram method and Maximum Likelihood (ML) type estimation [12]. The Histogram method depends on counting the sign changes between each two consecutive samples within one data bit interval, and comparing the result to two thresholds. The counters are reinitialized if two of them cross the lower threshold, and the bit edge is concluded if a counter reaches the upper threshold. The performance of this algorithm depends on the SNR and on the value of the two thresholds. ML synchronization calculates the average of the absolute sum of the correlator output over each consecutive one data bit interval samples, for each possible bit edge position, averages the sum over number of bit intervals, and chooses the edge that maximizes the sum. This technique gives high bit edge detection rate if there are no phase or frequency errors, but its performance degrades in their presence. Recently other methods also were developed, but are not discussed in this report. ML bit synchronization technique is chosen.

4.5 Data Decoding

The EGNOS data encoding mechanism is shown in Figure 4.4. The error correction code used is the $R = 1/2$, convolutional code with constraint length 7. This code introduces a decoding delay of 5 constraint lengths or 35 databits. It is possible to use MATLAB function for decoding EGNOS data. Viterbi's algorithm is well known in communication theory and it is a standard function of MATLAB Communication Toolbox.

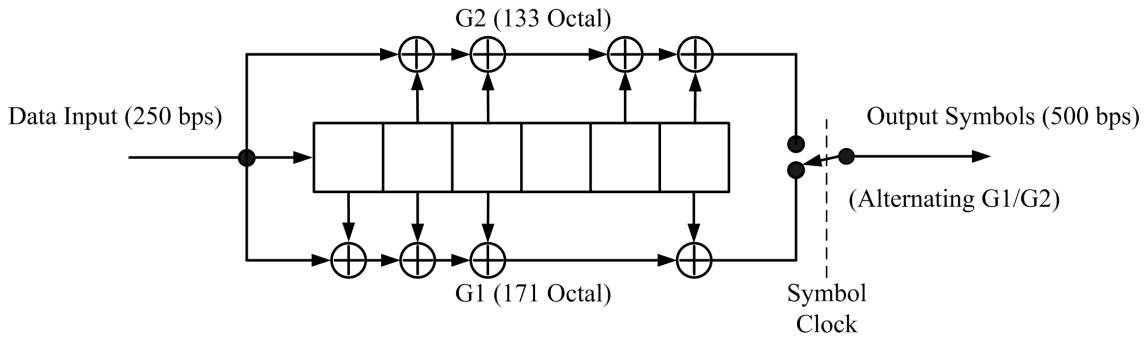


Figure 4.4. EGNOS data convolutional encoder

4.6 Processing EGNOS Data

The EGNOS structure consists of many messages, which are transmitted every second. After processing these messages, there will be updates of the system, which will let user to correct his position obtained by GPS receiver. The update consists of four error corrections (fast, long-term, ionospheric and tropospheric) and of quality information which can describe how good the processed data is in general.

4.6.1 Technical details

Message structure

The raw EGNOS message contains 500 bits, transmitted in each second. As it was mentioned before it is 1/2 convolutional encoded with a Forward Error Correcting (FEC) code. After decoding the message contains 250 bits of data. The structure of the message (Figure 4.5):

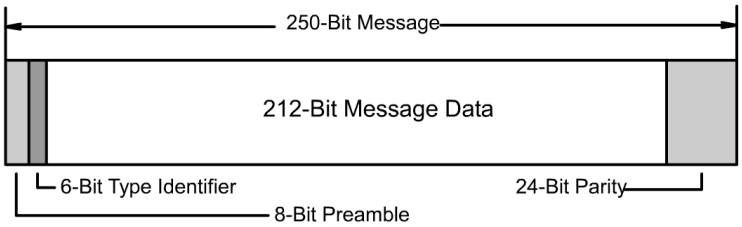


Figure 4.5. EGNOS message format

The preamble

The preamble is a 24-bit unique word, distributed in three subsequent messages. This field is needed for message synchronization and data extraction.

The message type identifier

Message type identifier is an integer number in a range from 0 to 63 which shows which type of the message was received.

The binary message

The message contains actual data which has to be decoded according to the message type identifier (Table D.2) [6]. The data is represented in unsigned binary format. In some cases, if data is represented in signed form, the two's complement decoding algorithm is used:

If first MSB (Most Significant Bit) bit of data value is equal to 1, then the number is negative:

$$Value = Value\_from\_the\_message - Resolution\_of\_the\_value$$

Else, if MSB bit is set to 0, the number is positive:

$$Value = Value\_from\_the\_message$$

Example:

$$Value\_from\_the\_message \quad = \quad 1001_2$$

then

$$Resolution\_of\_the\_value \quad = \quad 10000_2$$

and finally

$$Value = 1001_2 - 10000_2 \quad = \quad 9_{10} - 16_{10} = -7_{10}$$

Cyclic Redundancy Check (CRC) parity

CRC provides protection against burst and random errors [6]. The CRC word is calculated in the forward direction on the entire bit-oriented message including preamble, message identifier and binary message (0-225 bits). CRC sequence of 24 bits (p_1,p_2, ... ,p_{24}) is generated from the sequence of information bits (m_1,m_2, ... ,m_{226}) by using polynomial $g(X)$, which is called generator polynomial. $g(X)$ is given in the following form [6]:

$$g(X) = (1 + X)p(X) \tag{4.5}$$

where $p(X)$ is the primitive and irreducible polynomial:

$$p(X) = X^{23} + X^{17} + X^{13} + X^{12} + X^{11} + X^9 + X^8 + X^7 + X^5 + X^3 + 1 \tag{4.6}$$

then $g(X)$ can be expressed as:

$$g(X) = \sum_{i=0}^{24} g_i X^i \tag{4.7}$$

where $g_i = 1$ for $i = 0, 1, 3, 4, 5, 6, 7, 10, 11, 14, 17, 18, 23, 24$ and $g_i = 0$ otherwise.

The data is checked for errors by dividing $m(X)X^{24}$ by $g(X)$, where information sequence $m(X)$ is expressed as [6] [16]:

$$m(X) = m_k + m_{k-1}X + m_{k-2}X^2 + ... + m_1X^{k-1} \tag{4.8}$$

The result is a quotient and a remainder $R(X)$. The reminder represents the parity check sequence (p_1,p_2, ... ,p_{24}). If $R(X)$ is equal to that sequence, then the received code is correct, otherwise - the code has errors. In short we can write:

$$\left(\frac{\mathrm{m(X)X^{24}}}{\mathrm{g(X)}}\right)_{\mathrm{mod}} (\mathrm{p_1, p_2,..., p_{24}}) \quad = 0 \quad => \textit{no error}$$

$$\left(\frac{\mathrm{m(X)X^{24}}}{\mathrm{g(X)}}\right)_{\mathrm{mod}} (\mathrm{p_1, p_2,..., p_{24}}) \quad != 0 \quad => \textit{error}$$

Note: Arithmetic is done using binary polynomial algebra

Relation between message types

In order to relate data in different messages and to produce reliable solution all the messages have so called issue of data (IOD) parameters. It is important to track all these parameters when applying corrections, because small deviations can make dramatic changes in the output. The IOD parameters used in this project are (Figure 4.6):

1. *IOD* PRN mask (*IODP*) - the most important parameter which identifies the current PRN mask. The parameter is set when message type 1 has been received. Then, this parameter relates messages type 2-5, 24 (fast corrections) and 24-25 (long-term corrections) to the currently available PRN mask.

2. *IOD* Fast Corrections ($IODF_j$) - identifies the current fast corrections, where j shows fast corrections message type (2-5). This parameter is used to associate fast corrections with integrity information from message type 6.

3. *IOD* Ionospheric Grid Point Mask (*IODI*) - identifies the current Ionospheric Grid Point mask. This parameter relates Ionospheric Grid Point mask from message type 18 with actual Ionospheric delay corrections in message type 26.

4. GPS *IOD* Ephemeris ($IODE_k$) - indicates GPS ephemeris issue of data where k is satellite number. This parameter is used to associate broadcast GPS satellite ephemeris with long-term corrections. *IODE* in broadcast ephemeris should match the *IODE* value received with long-term corrections message. This is the only parameter which relates EGNOS message parameters with data "outside" EGNOS, thus, the synchronization should be kept to avoid errors.

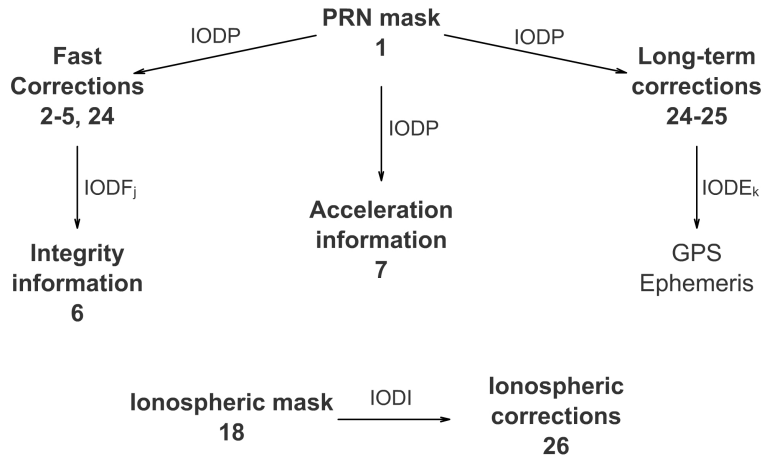


Figure 4.6. Interrelationships of messages

The *IOD* parameters are also used for so called alert conditions, which warn users about special situations, where they should act immediately in order to fix the emerged problems.

As it was described before, the messages are closely related between each other, but the message reception order is absolutely unknown. Any second one can expect any message type to be received. The only way to understand the concept of message transmission is by analyzing message maximum update intervals, which are different for each message type. For example,

message type 6 is kept the most important message, which provides integrity data and should be updated in 6 seconds. Other messages are less important, but still maintain defined update intervals as described in WAAS interface document [6]. Unfortunately it does not help to keep the system stable and under control. There are more different conditions which can lead to the situations where the message order can become different from the one, which is designed to be or in other words, the messages can be lost in transmission. This happens because of SBAS satellite shadowing or accidental bit transition errors. As it was mentioned before, if you do not know the message order and, despite that, you also lost some of the messages, then the output of the system can degrade and even without your knowledge. To prevent this, one should follow interface document [6] carefully and interpret it in correct way. Moreover, the system will work stable only if all the cases will be foreseen and implemented correctly.

4.6.2 Error corrections

Message type 1 - mask

Message type 1 is used to transmit PRN mask for all satellites monitored by EGNOS system. That includes 37 GPS satellites, 24 GLONASS satellites, 58 future GNSS satellites, 19 GEO satellites and 72 future GNSS/GEO/WAAS/Pseudolites.

The PRN mask shows which of the above mentioned 210 satellites will be used in EGNOS messages. Currently the total amount of satellites which can be used is reduced to 51. Monitored satellites in this mask have value set to 1, while unmonitored satellites have value set to 0. Additionally, the satellites for which corrections are provided must be ordered from 1 to 51 to decode messages type 2-5, 6, 7, 24 and 25. Then, ordered satellites will have special parameter called sequence number, which will show the satellite number in ordered sequence[1]. The Figure 4.7 shows how the mask is used and how sequence number is assigned.

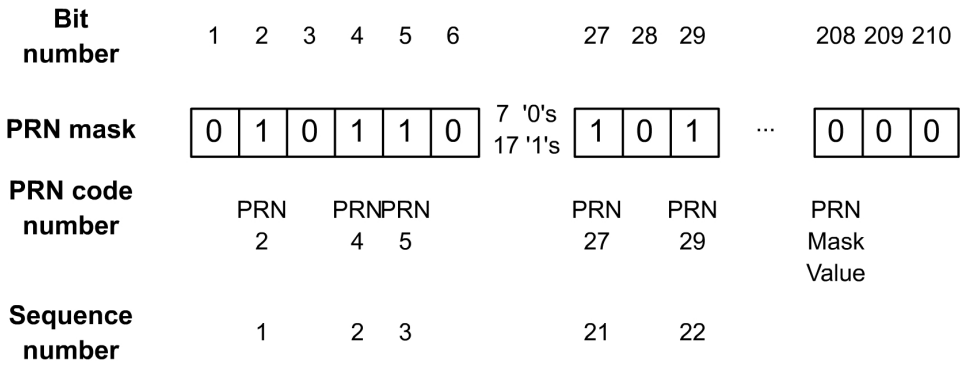


Figure 4.7. Example of PRN mask

[1]In this project only GPS satellites will be used, so the sequence numbers will be from 1 to 37

The message also has 2 bit $IODP$ value, which indicates the mask's applicability to the corrections and accuracies contained in messages to which the mask applies. If there are any changes in satellite constellations, then EGNOS will change the PRN mask as well as $IODP$ value will show this transition, which should happen infrequently. If such transition happened, then some precautions should be taken to ensure that new PRN mask agrees with transmitted corrections.

All messages in EGNOS have maximum update intervals and time-out conditions. For message type 1 the time-out interval is set to 600 seconds and maximum update interval is 120 seconds. That means that if this message is not received in 600 seconds, then the mask should not be used and, thus, corrections should not be applied. If one has just started to collect correction data and want to apply corrections, then he should wait for message type 1 in first place before collecting other data. That can take up to 2 minutes which we can call maximum initialization time for EGNOS corrections[2].

Fast error corrections

Fast corrections are necessary to correct the fast changing errors such as satellite clock error due to the (now no longer existing) degradation through Selective Availability (SA).

Information about fast corrections and related information is received with messages type 2-5, 6, 7 and 24 (Tables D.1, D.3 and D.5). The message types 2, 3, 4 and 5 have the same format and include 13 range corrections and 13 User Differential Range Error Indicators (UDREI) (Figure 4.8). The corrections are used to compute pseudorange corrections while the UDREI is the quality index, which is used to compute warning flags indicating that an individual PRN should not be used in the position solution [17]. To support corrections and UDREI values for all satellites, the data is distributed along all the messages, where first 13 satellites[3] ordered by sequence number is assigned to message type 2, next 13 satellites are assigned to message type 3 and so on, until message type 5, which has corrections for satellites from 40 to 51. One should mention that if there are less then 51 satellites, then some of the messages may not be transmitted. For example, if there are 33 satellites monitored by EGNOS, then message type 5 will not be transmitted.

More interesting is message type 24. This message contains fast corrections as well as long-term corrections. If one of the message types 2-5 has 6 or fewer satellites defined, then this message can be replaced by message type 24, where the first half of it is designed to provide 6 fast corrections and UDREI's. For example, if number of satellites used is between 40 and 45, then the message type 5 can be replaced by message type 24. This is done in order to optimize data transmission. To control corrections in message type 24 there is a parameter called Block ID, which can have values from 0 to 3 and indicates whether the message type 24 contains

[2]This is not valid for Ionospheric and Tropospheric corrections which are not using PRN mask from message type 1

[3]All the fast correction messages are related to the $IODP$ parameter which is used to assign corrections to the satellites defined in the PRN mask from message type 1

corrections associated with messages type 2, 3, 4 or 5 respectively (Figure 4.8).

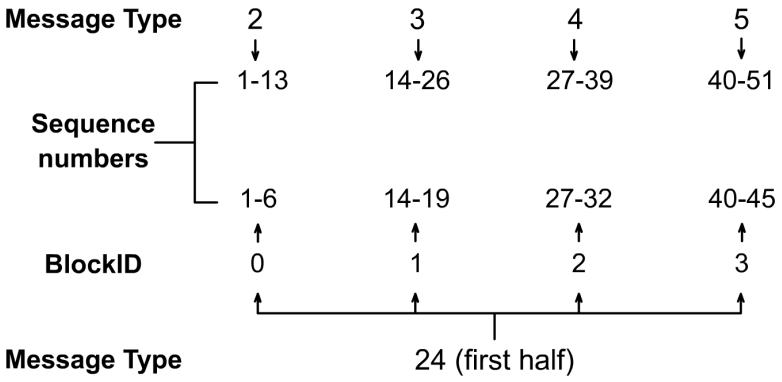


Figure 4.8. Satellite assignment in fast correction messages 2-5 and 24

UDREI indices are very important parameters and are used together with fast and long-term corrections to validate the quality of received information. UDREI values can be in a range from 0 to 15, where 14-15 indicate that given satellite is "not monitored" or signed to be not used ("do not use") respectively. According to some sources [17] the satellite corrections should not be used if UDREI values are higher than 12 for precision approach.

It is necessary that UDREI indices can be transmitted as often as possible to provide integrity information so the same UDREI indices are inserted in message type 6, which is called Integrity Information message and has UDREI values for all 51 satellites in one message. It is designed to be updated in less than 6 seconds (maximum update interval). This is done in order to warn user in 6 seconds about any changes in the system which has to be immediately adjusted.

To relate information from message type 2, 3, 4, 5 and message type 6 there is a parameter called $IODF_k$ (k - message type number (2-5)), which can have values from 0 to 3. Usually, if there is no alert condition (described below) $IODF_k$ can have values from 0 to 2, which will change with each new set of message arrival. This let one to synchronize and update the UDREI values received in message type 6 with messages 2, 3, 4, 5 and 24. Moreover, as the $IODF_k$ can cycle in a range from 0 to 2, the message type 6 contains UDREI values assigned only for one particular $IODF_k$. In this case, if we want to update all the satellites with different $IODF_k$ values, we have to receive at least 3 messages type 6.

If $IODF_k$ is received with value 3, then this situation is called alert condition. In this case all the $IODF_k$ values are ignored and message type 6 will update all the UDREI values for all satellites without excluding some satellites which may have different $IODF_k$ values than 3. This is done if there is a need to update information for more than one set of data in a very short time.

Pseudorange corrections (PRC) are updated using a certain interval which can vary from 6 seconds up to 60 seconds. This depends on the situation and is provided by SBAS service provider

using message type 7. This message contains fast correction degradation factor indicators (ai) (Table D.5) for all 51 satellites defined by PRN mask. Fast correction update interval shows that user will have PRC corrections every few seconds, while fast corrections are changing rapidly and user has to update pseudoranges every second. To fix this situation there is a range-rate correction (RRC) parameter introduced, which has to be computed by user. RRC is used to interpolate correction values in future using previously received PRC values. The concept of this is shown in Figure 4.9.

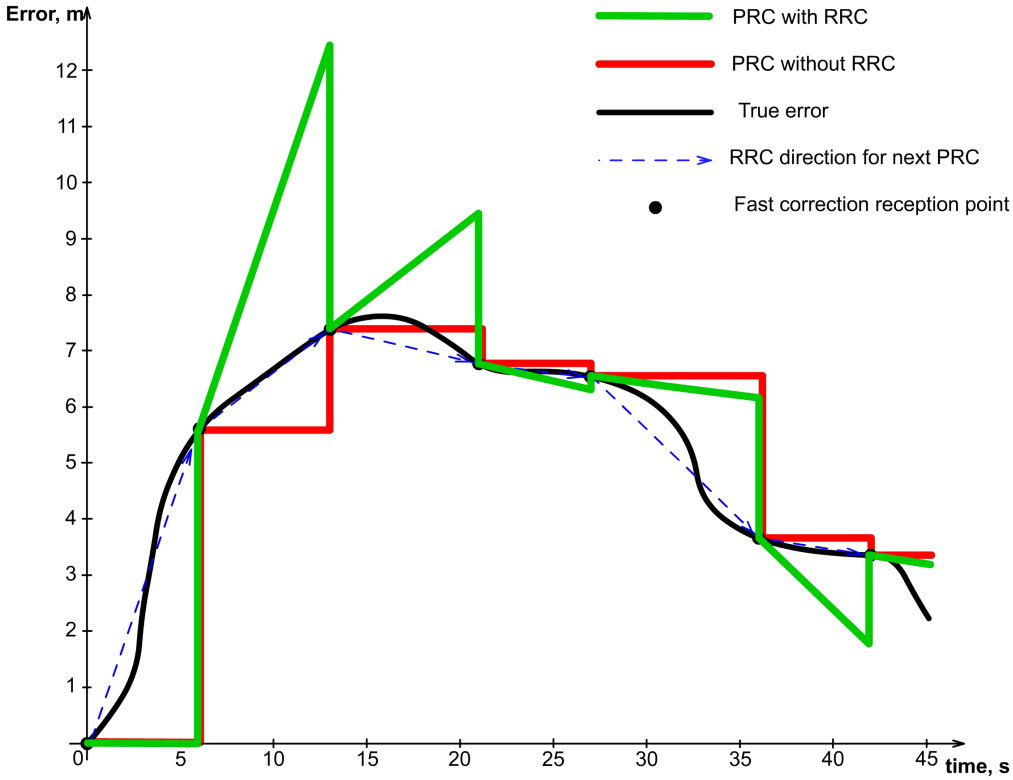


Figure 4.9. RRC computation and application

RRC is computed using the formula [6]:

$$RRC(t_{of}) = \frac{PRC_{current} - PRC_{previous}}{t_{of,current} - t_{of,previous}} \tag{4.9}$$

where

t_{of}	-	time of application
$PRC_{current}$	-	the latest received PRC value
$PRC_{previous}$	-	the PRC value received before $PRC_{current}$
$t_{of,current}$	-	the time when $PRC_{current}$ correction was received
$t_{of,previous}$	-	the time when $PRC_{previous}$ correction was received

However, RRC parameter computation has some restrictions. If the fast correction degradation parameter ai for a particular satellite is set to zero, then RRC shall be zero as well. Moreover, there are 3 time-out conditions defined for RRC parameter:

1. $RRC = 0$, if $(t_{of} - t_{of,previous}) > If_{c,j}$ (refer Table D.6)
2. $RRC = 0$, if $(t - t_{of-1}) > 8 \cdot (t_{of} - t_{of,previous})$
3. RRC - reinitialization, if UDREI with values 14 or 15 were received for some time and then shortly after this the UDREI values changed to UDREI < 14.

Finally, the total fast correction for a given satellite will be applied as:

$$RC_{fast} = PRC(t_{of}) + RRC(t_{of}) \cdot (t - t_{of}) \tag{4.10}$$

where

RC_{fast}	-	value to add to the measured pseudorange
t	-	time of application
t_{of}	-	time, when the most recent PRC value was received

Long-term error corrections

The purpose of the long-term corrections is to correct slowly varying errors such as the satellite position errors which arise in time because of uncertainties in satellite broadcast ephemeris. Another error source which is corrected by long-term corrections is the update of the satellite clock error, which is combined with clock error correction computed from ephemeris [17].

The long-term corrections are provided in the message type 25 and in the second half of message type 24. Message type 25 can transmit correction data for a minimum of 1 and a maximum of 4 satellites. The number of satellites transmitted in one message depends on some parameters inside the message. The message is divided in two independent parts and each of it has parameter *velocity code* (Tables D.7 and D.8) [6]. If this parameter is set to 0, then the indicated part will have data for two satellites and if velocity code is set to 1, then there will be data only for one satellite (Table 4.3).

In message type 24 there is only one part assigned for long-term corrections, so this message can contain correction data for up to 2 satellites.

Message Type 25	
1_{st} part	**2_{nd} part**
(Velocity Code = 0)	(Velocity Code = 1)
Data for Satellite 1 Data for Satellite 2	Data for Satellite 3

Table 4.3. Message type 25

Velocity code in long-term messages defines if there will be drift values (i.e. satellite velocity corrections and satellite clock drift corrections) (Table D.8) transmitted or not.

To assign long-term corrections to the satellite the $IODP$ parameter and PRN mask sequence number is used. However, the order of the satellites in PRN mask does not match the order in which the corrections are transmitted. Error corrections for satellites with faster changing long-term errors can be repeated at a higher rate than one with slower changing long-term errors.

To correct satellite clock error one should use the following formula [6]:

$$\delta\Delta t_{SV}(t_k) = \delta a_{f0} + \delta a_{f1} \cdot (t_k - t_0) \tag{4.11}$$

where

δa_{f0}	-	clock offset error correction (from the message)
δa_{f1}	-	clock drift error correction (from the message)[4]
$\delta\Delta t_{SV}$	-	satellite clock time error estimate
t_k	-	application time expressed as seconds-of-day[5]
t_0	-	time-of-day applicability (from the message)

To apply correction, the computed value should be expressed in meters:

$$RC_{clock} = \delta\Delta t_{SV}(t_k) \cdot c \tag{4.12}$$

Where c is the speed of light.

To correct satellite position error one should use this formula [6]:

[4] If velocity code is set to 0, then the value is also set to 0

[5] The time should be expressed in "seconds-of-day" whereas the user time is (in most cases) expressed as "seconds-of-week". Before any computations this should be corrected

$$\begin{bmatrix} \Delta X_{sv} \\ \Delta Y_{sv} \\ \Delta Z_{sv} \end{bmatrix} = \begin{bmatrix} \delta x \\ \delta y \\ \delta z \end{bmatrix} + \begin{bmatrix} \delta \dot{x} \\ \delta \dot{y} \\ \delta \dot{z} \end{bmatrix} \cdot (t_k - t_0) \tag{4.13}$$

where

$[\Delta X_{sv}, \Delta Y_{sv}, \Delta Z_{sv}]^T$	-	a vector, which will be added to the satellite position
$[\delta x, \delta y, \delta z]^T$	-	satellite position error (from the message)
$[\delta \dot{x}, \delta \dot{y}, \delta \dot{z}]^T$	-	satellite position error rate of change (from the message)[4]
t_k	-	application time expressed as seconds-of-day[5]
t_0	-	time-of-day applicability (from the message)

To apply long-term corrections one should check that the most up to date ephemeris be used for satellite position calculation. For this purpose the long-term messages include Issue of Data Ephemeris ($IODE$) value, which has to be checked with the one in ephemeris. If these values do not match, then corrections can not be used. Moreover, the user should encounter the cases when ephemeris is being updated. In such a case, when new ephemeris is uploaded into satellite, EGNOS will transmit long-term corrections still valid for the old ephemeris. This will continue for 2 minutes. This delay will enable the user to acquire the new GPS ephemeris data.

The long-term corrections and fast corrections are tightly related and if long-term corrections by any reasons can not be applied, then fast corrections also should not be used in computations and the particular satellite used in computations should be deselected from position computation [17].

Ionosphere error corrections

In contrast to the calculation of the fast and long-term corrections, the computation of ionosphere delay errors is more complex. The fast corrections and the long-term corrections are computed in straight forward way, while the ionospheric correction is computed using several steps.

The ionospheric delay corrections are broadcast as vertical delay estimates at specified ionospheric grid points (IGPs) [6]. In order to estimate corrections properly IGP points should widely cover the whole area supported by SBAS service provider. That corresponds to the large number of possible IGPs used. The predefined IGPs are spread into 11 bands for the whole world, where bands 0-8 are vertical bands on a Mercator projection map, and bands 9-10 are horizontal bands [6] (Figure 4.10). The density of these bands is defined according to the solar activity. As the grid of IGPs is very large, there are no possibilities to use it all at once. To overcome such problem, the SBAS provider will not use all the bands defined and will broadcast the mask, which will specify the area, where the model of the ionosphere is the most efficient at that time. EGNOS currently is using only the bands from 3 to 6[6], which entirely covers Europe

[6]It is unknown if EGNOS supports the horizontal bands 9-10

and some other regions.

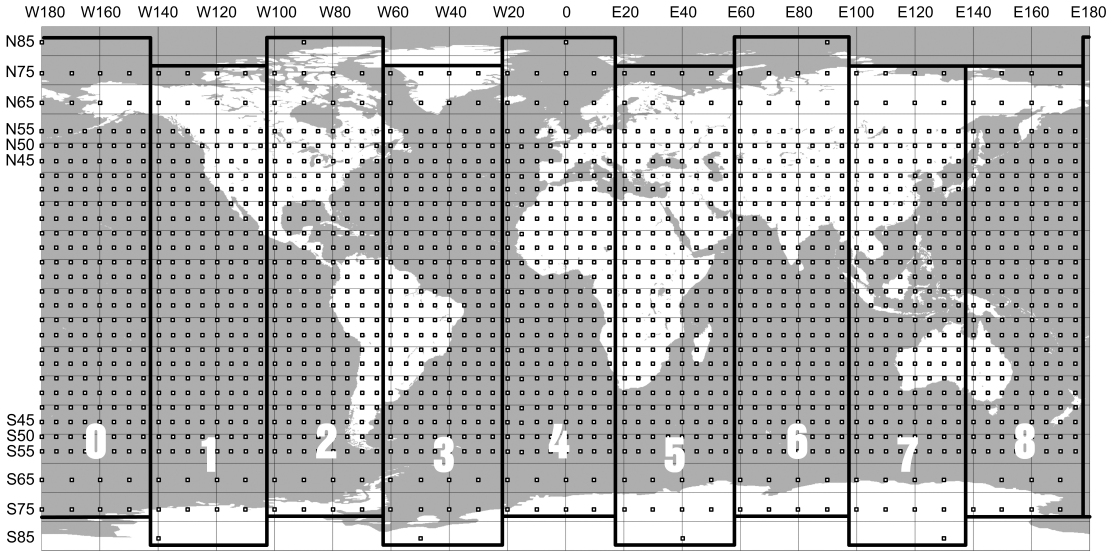


Figure 4.10. Predefined global IGP grid

The first step in ionosphere delay error computation is to obtain the ionospheric grid mask, which is transmitted in the message type 18 (Table D.10). As it was mentioned before, EGNOS transmit corrections only for 4 bands (Table D.9). To receive the mask, which will cover entire area one should receive 4 messages type 18 (one band per message). The grid mask is then stored permanently, but before that user should know, that mask is not distributed equally along the area. As the distance represented by a degree of longitude becomes smaller at higher latitudes, the IGP locations have different spacing at different latitudes. The IGP grid at the equator has 5° spacing, then to the north of $N55°$ and to the south of $S55°$ spacing is increased to 10° and finally at the latitudes of $N85°$ and $S85°$ the spacing is 90° (Table 4.4). Each band has 201 mask bits and covers 40° of longitude.

After user acquires the mask, the interpolation values of each defined grid point can be obtained through the message type 26 (Table D.11). It provides vertical delays relative to L1 signal and their accuracy parameters [6]. The received data is saved in the same format as the mask and the order is controlled by parameters *band number* and *block ID* which are received in message type 26 (Figure 4.11). Moreover, the data in message type 18 and 26 should be synchronized using $IODI$ values, which should match in both messages.

One message can contain only 15 data sets for 15 IGPs defined in the grid point mask and it can take some time before user can have vertical delay values for his location. To compute the quantity of messages required to receive all the data for defined bands the formula can be used:

Latitudes Degrees	Latitude Spacing Degrees	Longitude Spacing Degrees
N85	10	90
N75 to N65	10	10
S55 to N55	5	5
S75 to S65	10	10
S85	10	90 (Offset 40^o East)

Table 4.4. Predefined world-wide IGP spacing - bands 0-8

$$Message\_quantity = (201 \cdot bands\_quantity)/15 \approx 54$$

Where *bands_quantity* is the number of bands used (4 in this case).

The computed message quantity in reality never happens because the region covered by EGNOS is much smaller than defined by 4 bands. Moreover, the EGNOS will not transmit any corrections for the southern hemisphere, which will reduce the computed message quantity to the maximum of 27 messages (divide by 2).

The EGNOS system is designed to provide vertical delays for the grid mask with values set to 1 (Figure 4.11). This will further reduce time needed to receive data for user's location. The time in this case can vary from few seconds up to few minutes.

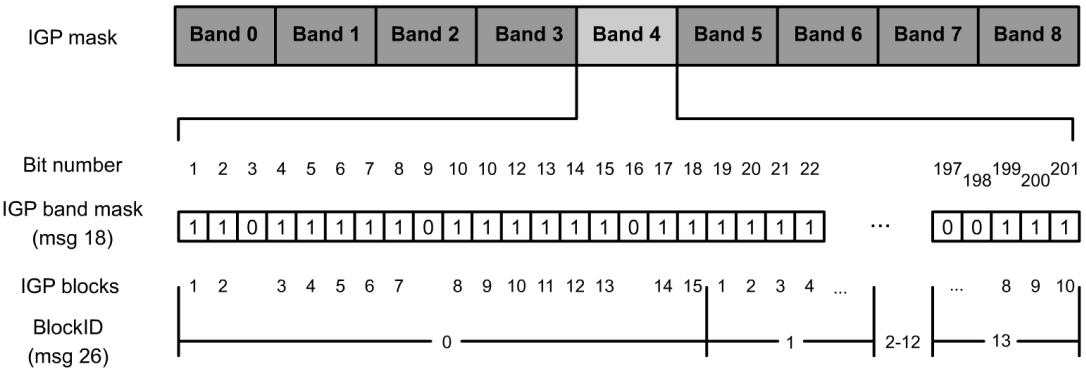


Figure 4.11. Example of ionospheric grid mask

If there are enough data received for user's area, the ionospheric pierce point has to be computed for each satellite. The ionospheric pierce point (IPP) is defined to be the intersection of the line segment from the receiver to the satellite and an ellipsoid with constant height of

350 km above the WGS-84 ellipsoid [6]. IPP is defined by longitude and latitude (Figure 4.12), which can be computed as [6]:

$$\phi_{pp} = \sin^{-1}\left(\sin\phi_u \cdot \cos\psi_{pp} + \cos\phi_u \cdot \sin\psi_{pp} \cdot \cos A\right)_{radians} \tag{4.14}$$

Where

ϕ_{pp} - pierce point latitude (in radians)
ϕ_u - user latitude (in radians)
A - azimuth of satellite (in radians)
ψ_{pp} - central earth angle (in radians)

ψ_{pp} is the earth's central angle between the user position and the earth projection of the pierce point computed as:

$$\Psi_{pp} = \frac{\pi}{2} - E - \sin^{-1}\left(\frac{\mathrm{R}_e}{\mathrm{R}_e + h_I}\cos E\right)_{radians} \tag{4.15}$$

Where

R_e - earth radius (assumed to be 6378.1363 km)
E - elevation of satellite (in radians)
h_I - height of maximum electron density (assumed to be 350 km)

Longitude:

$$\lambda_{pp} = \lambda_u + \pi - \sin^{-1}\left(\frac{\sin\Psi_{pp} \cdot \sin A}{\cos\phi_{pp}}\right)_{radians} \tag{4.16}$$

Where

λ_{pp} - pierce point longitude (in radians)
λ_u - user longitude (in radians)

After computing IPP the user has to select the IGPs to be used to interpolate the ionospheric correction. The selection is done according to the user location and the IGPs density (refer Appendix D.2 for more details). Moreover, the number of selected points around the IPP can differ from 3 to 4 (depends on available vertical delays in the area).

Selected vertical IGP delays do not generally correspond with previously computed IPP location, so the user have to interpolate these values. According to the number of surrounding vertical IGP delay points selected, the user has to use different formulas. For four-point interpolation user have to use the following formula:

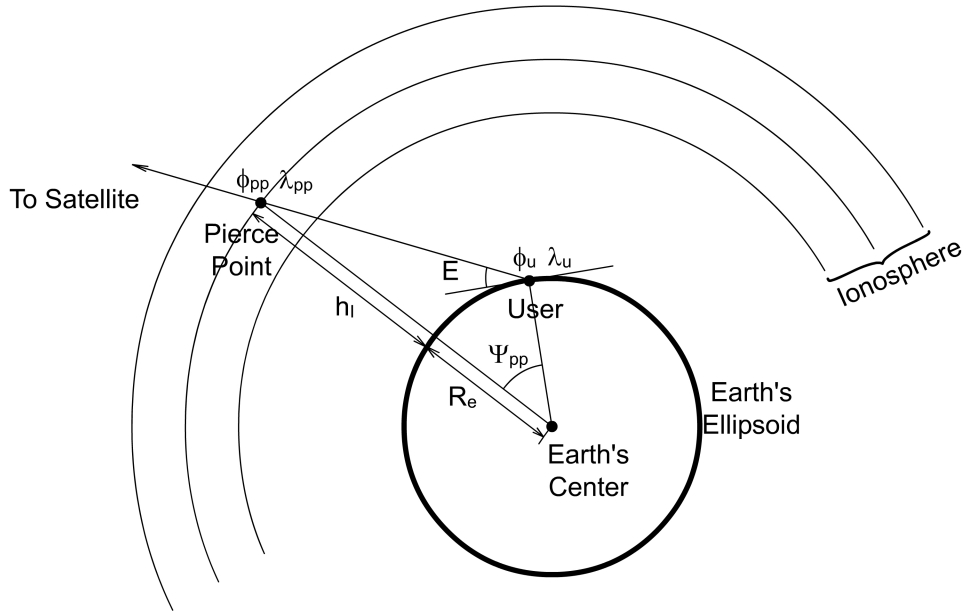


Figure 4.12. Pierce point computation

$$\tau_{vpp}\left(\phi_{pp},\lambda_{pp}\right)=\sum_{i=1}^{4}W_i\left(x_{pp},y_{pp}\right)\cdot\tau_{vi} \tag{4.17}$$

Where

τ_{vpp}	-	interpolated vertical delay at defined IPP
ϕ_{pp}	-	IPP latitude
λ_{pp}	-	IPP longitude
W_i	-	Weighting function for four points (refer to D.3.1)
τ_{vi}	-	selected grid point vertical delay values at four corners of the IGP grid (refer to D.3.1)

Weighting function depends only on geometry of four selected grid points (refer Appendix D.3).

For three-point interpolation user have to use the following formula:

$$\tau_{vpp}\left(\phi_{pp},\lambda_{pp}\right)=\sum_{i=1}^{3}W_i(x_{pp},y_{pp})\tau_{vi} \tag{4.18}$$

Where

W_i - Weighting function for three points (refer to D.3.2)
τ_{vi} - selected grid point vertical delay values at tree corners of the IGP grid (refer to D.3.2)

The final ionospheric delay correction will be computed by multiplying the interpolated vertical delay value at the user pierce point by the obliquity factor F_{pp}:

$$RC_{iono} = -\tau_{spp}\left(\lambda_{pp}, \phi_{pp}\right) = -F_{pp} \cdot \tau_{vpp}\left(\lambda_{pp}, \phi_{pp}\right) \tag{4.19}$$

Where

τ_{vpp} - the interpolated vertical delay at pierce point
F_{pp} - obliquity factor

Obliquity factor is computed as:

$$F_{pp} = \left[1 - \left(\frac{\mathrm{R}_e \cdot \cos E}{\mathrm{R}_e + h_I}\right)^2\right]^{-\frac{1}{2}} \tag{4.20}$$

Troposphere error corrections

Troposphere delay error is the last correction proposed by EGNOS, but in general it is not part of the system. The tropospheric refraction is a local phenomenon and all users have to compute it according to their current location [6].

The computation of this correction is done in three steps:

1. The 5 meteorological parameters (pressure, temperature, water vapour pressure, temperature lapse rate and water vapour lapse rate) used in the troposphere correction model are taken from the table proposed by EGNOS and interpolated according to the current user latitude. For latitudes less than 15° or more than 75° the meteorological values are taken directly from the tables (refer Table D.13) and for latitudes in the range between 15° and 75°, the values are calculated using the following formulas:

$$\xi_0(\phi) = \xi_0(\phi_i) + \left[\xi_0(\phi_{i+1}) - \xi_0(\phi_i)\right] \cdot \frac{(\phi - \phi_i)}{(\phi_{i+1} - \phi_i)} \tag{4.21}$$

$$\Delta\xi_0(\phi) = \Delta\xi_0(\phi_i) + \left[\Delta\xi_0(\phi_{i+1}) - \Delta\xi_0(\phi_i)\right] \cdot \frac{(\phi - \phi_i)}{(\phi_{i+1} - \phi_i)} \tag{4.22}$$

Where

ξ_0	-	mean meteorological parameter
$\Delta\xi_0$	-	seasonal variation meteorological parameter
ϕ	-	user latitude
ϕ_i	-	latitude of first interpolation point
ϕ_{i+1}	-	latitude of second interpolation point

Then, the interpolation is done using the current date (day-of-year).

$$\xi(\phi, D) = \xi_0(\phi) - \Delta\xi(\phi) \cdot \cos\left(\frac{2 \cdot \pi \cdot (D - D_{min})}{365.25}\right) \tag{4.23}$$

Where

$\xi(\phi, D)$	-	actual meteorological parameter
$\xi_0(\phi)$	-	mean meteorological parameter (from Formula 4.21)
$\Delta\xi(\phi)$	-	seasonal variation of the meteorological parameter (from Formula 4.22)
D	-	day-of-year
D_{min}	-	constant offset specified as: $D_{min} = 28$ for northern latitudes, and $D_{min} = 211$ for southern latitudes

2. The zero-altitude zenith delay terms for the wet and the dry part of the error compensations are calculated. These computed delays are then modified to account for the user's altitude.

$$z_{hyd} = \frac{10^{-6} \cdot k_1 \cdot R_d \cdot P}{g_m} \tag{4.24}$$

$$z_{wet} = \frac{10^{-6} \cdot k_2 \cdot R_d}{g_m\,(\lambda + 1) - \beta \cdot R_d} \cdot \frac{e}{T} \tag{4.25}$$

Where

z_{hyd} - dry contribution to zero-altitude zenith delay
z_{wet} - wet contribution to zero-altitude zenith delay
P - meteorological pressure using equation $(\xi(\phi, D))$
e - water vapour pressure using equation $(\xi(\phi, D))$
T - temperature using equation $(\xi(\phi, D))$
β - temperature lapse rate using equation $(\xi(\phi, D))$
λ - water vapour lapse rate using equation $(\xi(\phi, D))$
k_1 - constant 77.604 $k/mbar$
k_2 - constant 382000 $K^2/mbar$
R_d - constant 287.054 $J/kg/K$
g_m - constant 9.784 m/s^2

The zero-altitude zenith delay must then be modified to take the user's altitude into account [17]. This is done by the following equations:

$$d_{hyd} = \left(1 - \frac{\beta \cdot H}{T}\right)^{\frac{g}{R_d \cdot \beta}} \cdot z_{hyd} \tag{4.26}$$

$$d_{wet} = \left(1 - \frac{\beta \cdot H}{T}\right)^{\frac{(\lambda+1) \cdot g}{R_d \cdot \beta} - 1} \cdot z_{wet} \tag{4.27}$$

Where

d_{hyd} - dry contribution to zenith delay
d_{wet} - wet contribution to zenith delay
H - user altitude expressed in meters above mean sea level (the procedure how to compute H is given in page 55)
g - constant 9.80665 m/s^2

3. The computed estimation is updated using satellite elevation angle transforming the estimation of the vertical delay into an estimation of the slant delay [17]. Moreover, the new alternative model is used to compute the obliquity factors [18] which improves algorithm performance at low satellite elevations. WAAS interface document [6] currently specifies the model, which is valid only for satellite elevations higher than 5 degrees, while the new model is valid for higher than 2 degrees satellite elevations. The obliquity factors for d_{hyd} and d_{wet} are computed as:

$$m_{hyd}, m_{wet} = \frac{1 + \frac{a}{1 + \frac{b}{1+c}}}{\sin(E_i) + \frac{a}{\sin(E_i) + \frac{b}{\sin(E_i)+c}}} \tag{4.28}$$

Where E_i is satellite elevation angle (in radians) and parameters for m_{hyd} and m_{wet} are computed in the following way:

For m_{hyd}:

$$a_{hyd} = (1.18972 - 0.026855 \cdot H + 0.10664 \cdot \cos(\phi)) \cdot 10^{-3}$$
$$b_{hyd} = 0.0035716$$
$$c_{hyd} = 0.082456$$

Parameters for m_{wet}:

$$a_{wet} = (0.61120 - 0.035348 \cdot H - 0.01526 \cdot \cos(\phi)) \cdot 10^{-3}$$
$$b_{wet} = 0.0018576$$
$$c_{wet} = 0.062741$$

Finally, the tropospheric delay estimate is computed as:

$$RC_{tropo} = -(d_{hyd} \cdot m_{hyd} + d_{wet} \cdot m_{wet}) \tag{4.29}$$

Position calculation and correction

In order to process EGNOS corrections the following data should be available (Figure 4.13):

1. Pseudoranges for all SV's
2. Valid satellite ephemeris for all SV's
3. GPS timestamp for supplied data
4. EGNOS message

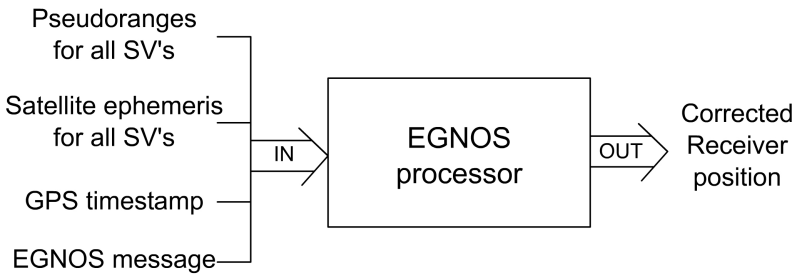


Figure 4.13. Planned system inputs and outputs

The supplied data can be divided into three groups:

1. **EGNOS corrections.** This data could be received using software EGNOS receiver or any other receiver capable to process EGNOS signals and provide binary EGNOS messages. According to the EGNOS system the position corrections should be computed once per second and, thus, the message should be supplied in the same interval.

2. **GPS data.** The data is provided by GPS receiver, which is capable to issue raw data. In order to apply corrections only the raw GPS measurements should be used. Else, the output performance can significantly degrade. The GPS data also should be supplied in one second interval[7].

3. **Timestamp.** In order to use EGNOS and GPS data the synchronization should be maintained. The EGNOS message does not contain any timestamp, thus it should be assigned by the receiver, which provided EGNOS message. Moreover, the timestamp should match[8] the one used to mark the data from GPS receiver.

The correction of raw GPS data is done by updating pseudorange measurements:

$$\rho = \rho_{meas} + RC_{fast} + RC_{iono} + RC_{tropo} + RC_{clock} \tag{4.30}$$

Where

ρ	-	the corrected pseudorange (in meters)
ρ_{meas}	-	the measured pseudorange
RC_{fast}	-	fast corrections (from Formula 4.10 page 43)
RC_{iono}	-	ionosphere delay corrections (from Formula 4.19 page 50)
RC_{tropo}	-	troposphere delay corrections (from Formula 4.29 page 53)
RC_{clock}	-	clock corrections for particular satellite (from Formula 4.12 page 44)

After the pseudoranges are corrected the satellite positions are computed as described in Appendix E.1. Then, the satellite positions are corrected:

$$\begin{bmatrix} X_{c,sv} \\ Y_{c,sv} \\ Z_{c,sv} \end{bmatrix} = \begin{bmatrix} X_{sv} \\ Y_{sv} \\ Z_{sv} \end{bmatrix} + \begin{bmatrix} \Delta X_{sv} \\ \Delta Y_{sv} \\ \Delta Z_{sv} \end{bmatrix} \tag{4.31}$$

Where

$[X_{c,sv}, Y_{c,sv}, Z_{c,sv}]^T$	-	corrected satellite position[9]
$[X_{sv}, Y_{sv}, Z_{sv}]^T$	-	raw satellite position
$[\Delta X_{sv}, \Delta Y_{sv}, \Delta Z_{sv}]^T$	-	satellite position correction (from Formula 4.13 page 45)

[7] Valid only for pseudoranges.Ephemeris should be supplied only during initialization or shortly after the satellites were updated with new ephemeris

[8] As it is not always possible to maintain perfect synchronization, the timestamp should be within the fraction of a second. The GPS data timestamp is used as the reference point

Finally, the receiver position is computed as described in Appendix E.2.
The output of the system is corrected receiver position, which is issued in every second.

4.6.3 Further EGNOS system analysis

This section describes EGNOS system differences from WAAS system as well as explains how to compute receiver height above mean sea level required in troposphere error computation.

EGNOS system limitations and differences comparing to WAAS

WAAS interface document [6] describes how entire SBAS system should work. WAAS, EGNOS and MSAS are Satellite Based Augmentation Systems (SBAS) and by the help of the standard all these systems can work together to provide global coverage and other advantages. In order to keep everything under the standard the EGNOS system should follow this document closely, but in some cases there are differences which you have to take into account:

1. **Message type 0** provides no information inside its body, but if transmitted, user should immediately stop data processing for the next 60 seconds from SBAS satellite which transmitted this message. This is valid for operational systems, but not for EGNOS, which is still in testing mode and thus, this should be disregarded in order to use EGNOS. The transmission of message type 0 in EGNOS means that no-one should use the system for safety applications. Moreover, there are differences in messages type 0 comparing EGNOS and WAAS systems. WAAS system during testing mode or during ordinary operation mode is transmitting message type 2 instead of message type 0 (message type 2 with message type 0 identifier). EGNOS transmit normal 0 type message all the time informing users that the system is still in testing mode. This limits EGNOS performance, because message type 0 comes very often (every 6 seconds) and limits the number of messages which can come instead of message type 0.

2. **Ionospheric grid point mask.** The different SBAS systems differ mostly because of location for which corrections are produced and transmitted. The main difference between systems lies in Ionospheric correction matrix used, defined as Ionospheric Grid Point Mask. The WAAS system is projected to support mainly North America users, while EGNOS target is Europe area only. Currently EGNOS system covers (not necessarily provide corrections) the area described as bands 3 to 6 (refer section 4.6.2 page 45).

Height transformation

The user position coordinates are computed in Earth-Centered, Earth-Fixed (ECEF) X, Y, Z coordinates. To compute position which we can show on the map it is enough to transform our three-dimensional coordinates into two-dimensional. The third coordinate, which in two-dimensional plane will represent the height of location, is not that important in our case.

However, to compute tropospheric delay error using the model proposed by WAAS MOPS document [6] one have to use height expressed in meters above mean sea level (MSL). The GPS system uses World Geodetic System 1984 (WGS-84) as a reference frame for position calculations (satellites and receiver) and in this system height is expressed as the height above WGS-84 ellipsoid (HAE). To use the correct height one should perform transformation between two systems. The required transformation will be explained using Figure 4.14. Here, the point A is user position. To convert the height from one system to another we use formula:

$$h_A = H_A - N_A$$

Where

h_A - height above mean sea level (MSL)
H_A - height above the WGS-84 ellipsoid (HAE)
N_A - the Geoid Separation (Undulation)

According to the formula for height transformation we will need only one parameter - N_A. To get it we have to use the detailed gravity model of the earth defined as Earth Gravity Model 1996 (EGM-96). The EGM-96 model consist of 0.25 degree grid of Geoid undulation values (Figure 4.15).

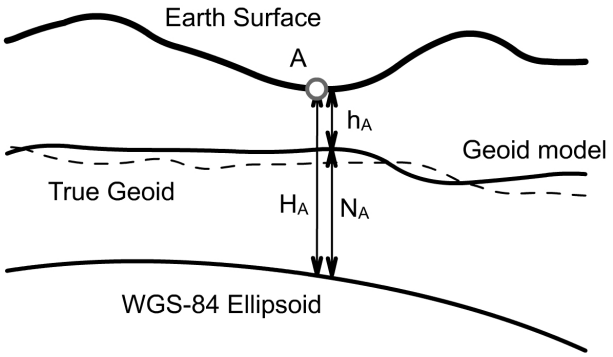


Figure 4.14. Difference between orthometric (MSL) and ellipsoidal (HAE) height

The geoid undulation value at user position is calculated by applying a correction term that converts a pseudo-height anomaly calculated at a point on the ellipsoid to a geoid undulation value [19].

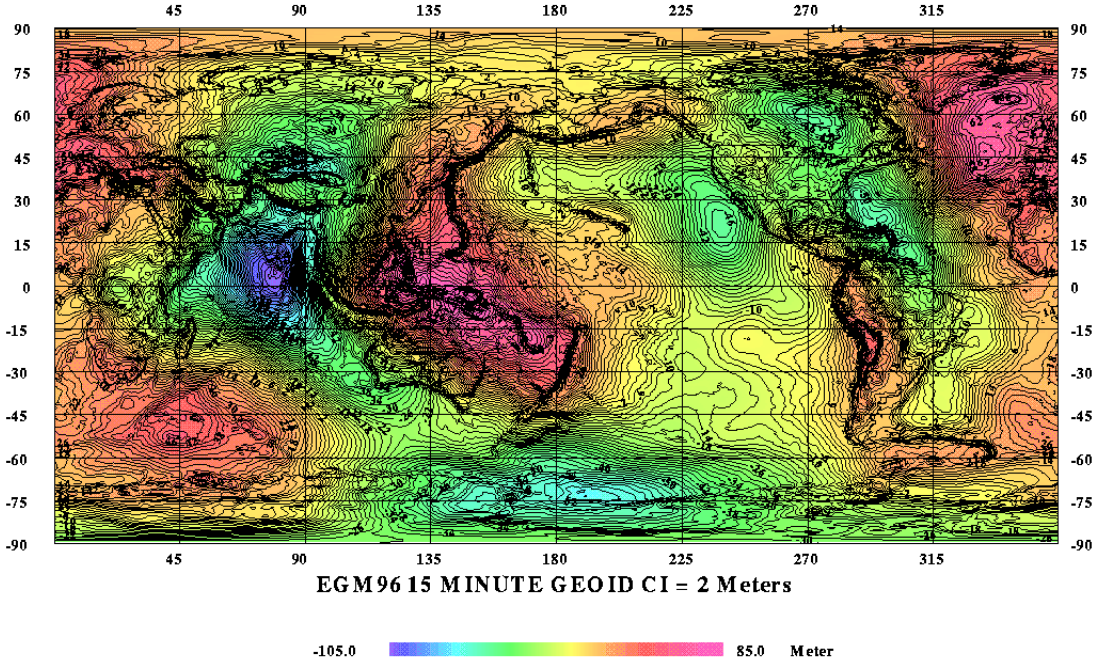


Figure 4.15. Earth Gravity Model (EGM-96)

4.7 Summary

An overview of general problem analysis relevant to EGNOS signal acquisition, tracking, data decoding and application of corrections was discussed. The EGNOS satellites have different PRN codes than GPS, although being from the family of gold codes, their acquisition requires a modified G2 st state coder. The acquisition requires search in the range of the uncertainty of code and carrier doppler. As the EGNOS satellites are geostationary the doppler effect is relatively smaller. The correlators are required to get the signal peaks by correlating with replica signals. Once the signal is acquired it should be tracked in order to make the incoming and replica signal coherent. Several tracking methodologies have been discussed along with bit synchronization issues. EGNOS data decoding message types and several error corrections have been discussed.

Chapter 5

Design and Implementation

The purpose of this chapter is to describe the algorithms that are used and analyze their performance.

5.1 Receiver Architecture

The simplified block diagram of the designed prototype EGNOS software receiver is shown in Figure 5.1.

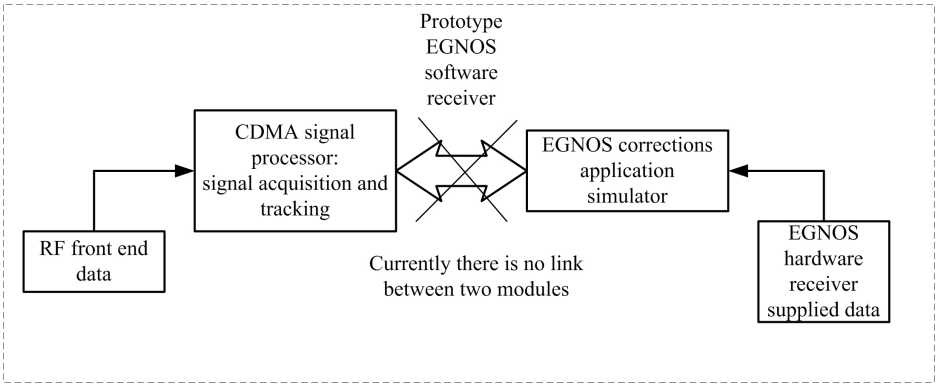


Figure 5.1. Receiver architecture

The first module is the CDMA signal processor, where signal acquisition and tracking schemes are realized. Acquisition is implemented in frequency domain, and tracking is performed in time domain. The FFT-based signal acquisition scheme is implemented. Second order Costas PLL and ordinary DLL are realized for signal tracking.

The second module is the EGNOS corrections simulator, where the EGNOS corrected GPS

position is computed. This module deals with GPS and EGNOS satellite messages data post-processing. The purpose of this module is to provide an improved civil GPS positioning solution.

The data for processing for modules is supplied from different sources. Currently there is no link between two modules and this will be the main issue of future project improvements. For this, additional problems with hardware matters required to be solved. As mentioned in chapter 4 the project was not dealing with RF front end hardware issues.

The design and implementation is described in the order of development stages, that were defined in chapter 3.

5.2 Signal Acquisition

Two steps are consisted in the acquisition process. The first is noise floor computation and coarse signal acquisition, and the second is carrier Doppler frequency refinement. The FFT-based noncoherent correlator is used. Coherent correlation time is defined to be 1 ms.

The acquisition implementation is illustrated in Figure 5.2.

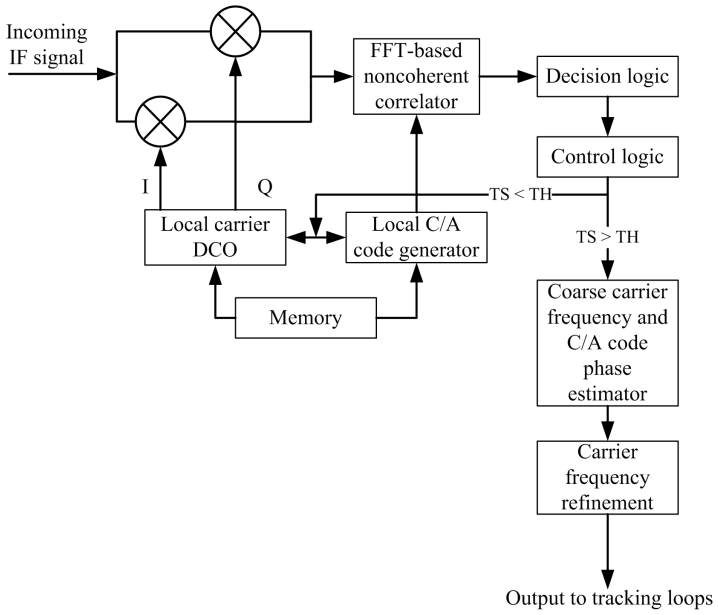


Figure 5.2. Implemented acquisition scheme

Coarse acquisition

The following steps are performed by coarse acquisition:

1. Search for the signal both in frequency and C/A-code phase.

2. Determine which satellites are visible, detect the presence of a signal and confirm detection.

3. Determine the approximate Doppler of each visible satellite.

The FFT search algorithm is used in MATLAB implementation. The incoming signal is mixed to baseband (that is multiplied with complex multiplier) and the in-phase (I) and quadrature (Q) components are used as the real and imaginary inputs when calculating the FFT. The result is multiplied by the complex conjugate of FFT of the C/A code. The circular convolution is obtained by taking the magnitude of the inverse FFT (see Figure 4.2).

A MATLAB function generates a correlation matrix R by performing the FFT search over the Doppler range and K sequential correlations. A test statistics TS is calculated by dividing the maximum value of R by the average value of all elements in R. This metric is tested against the threshold TH, and if TS>TH the satellite is assumed to be acquired. If TH<TS the satellite is assumed to be not present.

As K increases the TS changes (decreases), therefore the TH also should be variable. TH should be precomputed before acquisition process. For this acquisition noncoherent integration time K is maximally 10 ms. Thresholds (noise floor values) for every millisecond are experimentally estimated.

In order to avoid acquisition false alarm, an exhaustive frequency search is performed. Three FFT operations can complete the PRN code offset search at one frequency bin. Several methods were applied to further reduce the computational burden [20].

First, the input signal is common to all channels no matter which satellite is being acquired, so the FFT of the incoming signal after multiplying all possible local carriers can be conducted only once for every satellite. Obviously, the more the satellites under acquisition, the more saving can be achieved with the computational load. Second, the FFTs of the local C/A codes do not change, so they can be computed in advance and saved in a buffer. As the result, the number of FFTs required is reduced by one third.

A function for generating the C/A codes was coded in MATLAB. It produces the C/A code sequences, for each satellite with the PRN number as the input parameter. All generated codes were stored in memory. Local carrier replicas were precomputed and also stored in memory on a rough grid of 500Hz.

An acquisition plot of a GPS satellite on 1 ms of data is shown in Figure 5.3.

A plot, when a satellite signal is not present, is shown in Figure 5.4.

Acquisition of EGNOS satellite on 1ms data block is illustrated in Figure 5.5.

From the Figure 5.5 can be seen that EGNOS satellite signal is still buried in noise. The noncoherent integration time in such cases is increased. The acquisition of EGNOS satellite on 10 ms data is illustrated in Figure 5.6.

A considerable improvement can be observed. Now the satellite signal can be found.

Frequency refinement

An important issue to be solved in signal acquisition is frequency refinement.

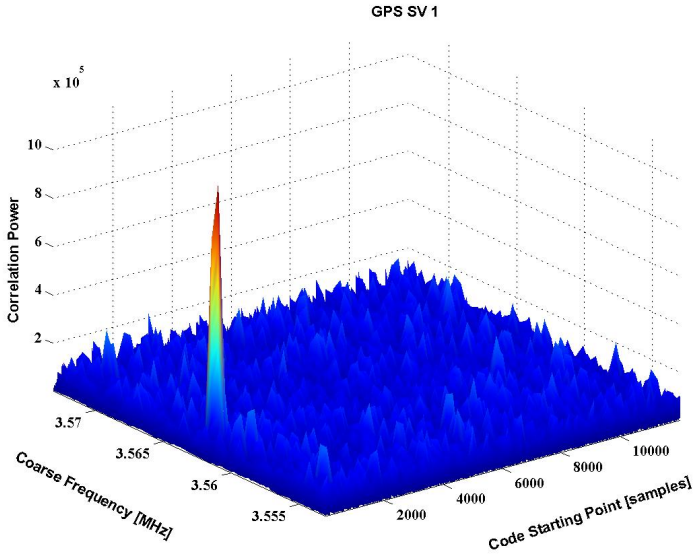


Figure 5.3. Acquisition of a GPS satellite

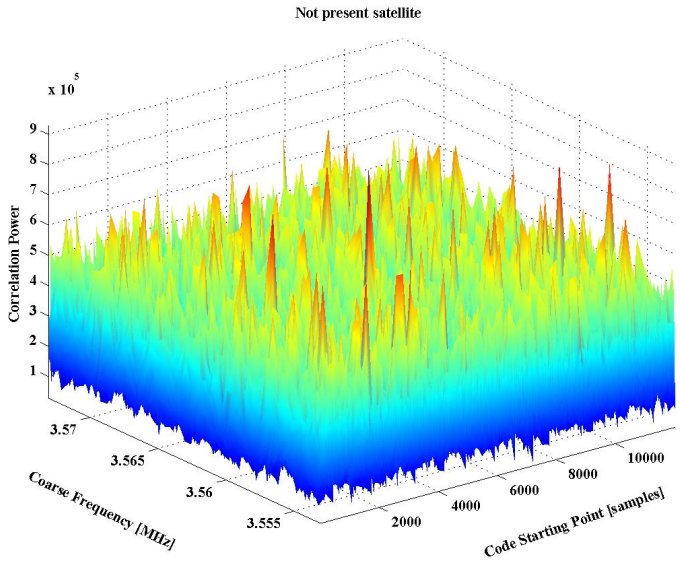


Figure 5.4. Noise, when a satellite signal is not present

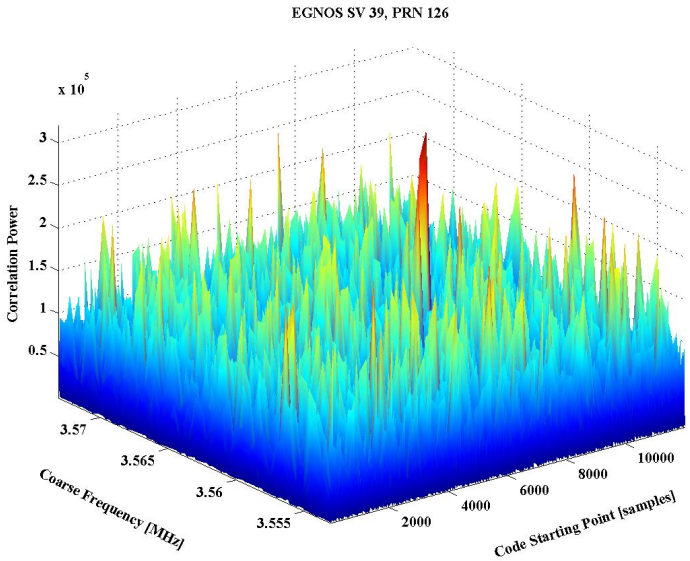


Figure 5.5. Acquisition of an EGNOS satellite, on 1ms

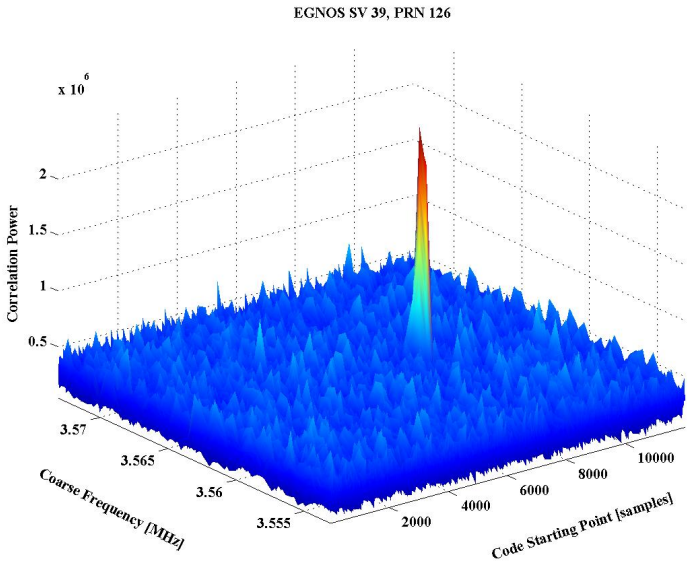


Figure 5.6. Acquisition of an EGNOS satellite, on 10 ms

An analytical frequency refinement method based on the shape of the correlation waveform is used as proposed by [20].

The signal amplitude in acquisition can be defined as:

$$G = \sqrt{I^2 + Q^2} \tag{5.1}$$

where

I - In-phase correlation component
Q - Quadrature correlation component

When the coherent integration time is 1 ms, G will have a normalized waveform as illustrated in Figure 5.7. Here, the horizontal axis represents the acquisition frequency error and the vertical axis represents the normalized signal amplitude. Three frequency bins, with 500Hz width each one, contain three correlation peaks. The highest peak in acquisition is in the central bin. Three searching points, denoted by L, P, and E. The signal amplitudes of the search point L, P, E are a function of frequency error. A discriminator can be constructed to derive such frequency error based on the measured amplitudes L, P, E.

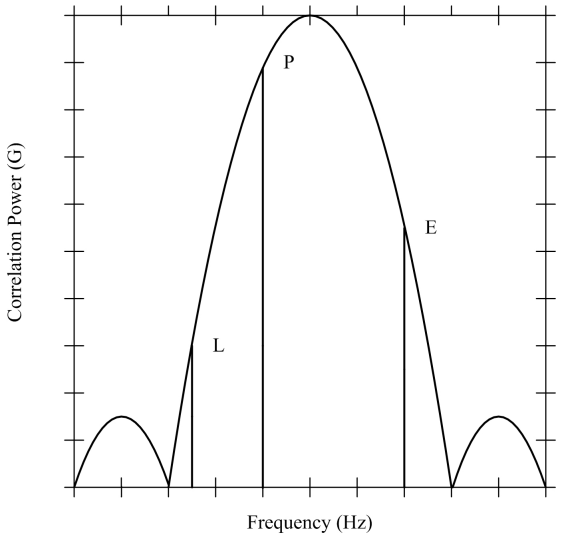


Figure 5.7. Normalized acquisition waveform in frequency domain

The following discriminator is used for frequency refinement:

$$F = 5 \cdot \frac{G_L - G_E}{G_P} \tag{5.2}$$

where

G_L - signal amplitude at L point
G_E - signal amplitude at E point
G_R - signal amplitude at R point

The discriminator is a function of the acquisition frequency error. The discriminator is approximated with the following third order polynomial:

$$m(F) = -4.135F^3 + 204.5F \tag{5.3}$$

where F is the discriminator output.

The final frequency estimate is derived by removing the frequency error from the coarse frequency estimate as follows:

$$f_{acq} = f_P + m(F) \tag{5.4}$$

where f_P is the estimated frequency from the coarse acquisition and $m(F)$ is the frequency correction calculated from the discriminator output.

Modified averaging correlator

In order to improve acquisition speed the modified averaging correlation method was implemented and tested.

The upsampled incoming and local signals are downsampled to a less sampling frequency of 4.096 MHz. In this case 1 ms of data becomes 4096 points. It is a radix 2 number. FTT computations of this number of points are efficient. However, since the signal is downsampled it is unknown where the exact local C/A starting point is in the downsampled signal. One C/A code chip is approximately 12 samples[1]. In order to find the correct starting point of the C/A code in the downsampled data it is necessary to perform 12 different correlations, every with different C/A code starting point. 12 times 4096 FFT is still cheaper than 11999 FFT, no matter in hardware or just software simulation in MATLAB.

In the case this method is only applied to rough acquisition, the correlation of one or two out of 12 correlation sequences with C/A code can already locate the right peak [13]. Only 1 of 12 correlations was implemented in MATLAB function for this method. Such implementation is only suitable for rough signals acquisitions with high SNRs.

A good point of this method is that it is not going to cause loss of signal energy compared with correlation by 11999 FFT.

An acquisition of EGNOS satellite by modified averaging correlator on 6 ms data is shown in Figure 5.8.

[1]The original sampling frequency is 11.999MHz

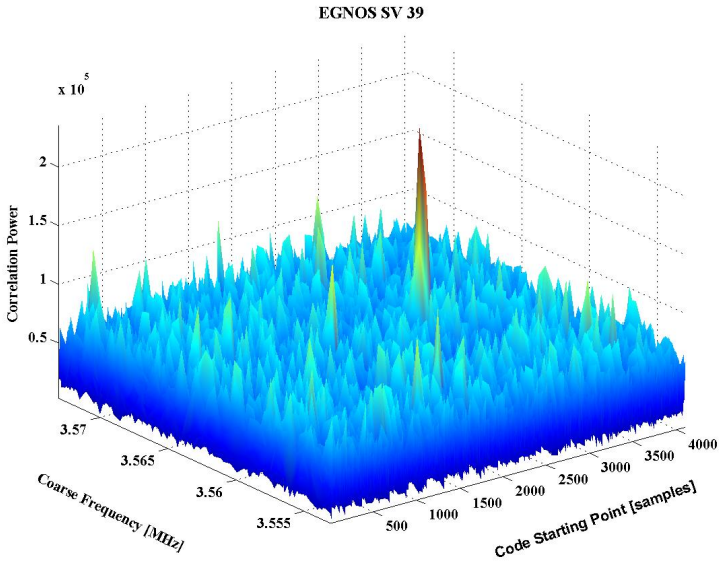


Figure 5.8. Acquisition of EGNOS satellite by modified averaging correlator

In the case of modified averaging correlator the code starting point is estimated with respect to downsampled signal. Therefore this estimate is transformed to a normal one like obtained in the case of normal FFT-based correlator.

This method was implemented only for testing purposes and was not used in the final version of acquisition module. However, it should be noticed that the acquisition time with modified averaging correlator was at least twice faster than the time of ordinary correlator. Thus, the usage of modified averaging correlator in the refined version of the receiver is preferable.

Once the acquisition of a signal has been achieved, the processing can proceed to a tracking mode in which two loops operate in parallel.

5.3 Signal Tracking

The following steps are performed by receiver tracking module:

1. Lock onto and track the C/A-code.

2. Lock onto and track the carrier.

The implemented tracking module schematics is represented in Figure 5.9
Ordinary PLL for carrier tracking, and DLL for code tracking are implemented.

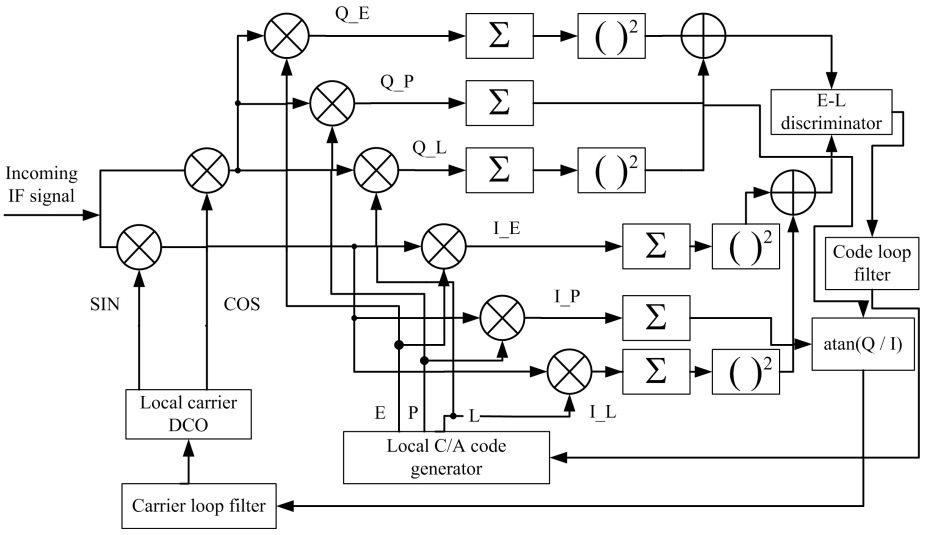


Figure 5.9. Tracking module schematics

The implemented DLL consists of early, prompt and late code generators, filter and a discriminator. The early and late codes are obtained by shifting a prompt code in time by 5 samples, that is approximately half a C/A code chip. The early and late codes correlate with incoming C/A codes to produce two outputs. These outputs are filtered (averaged), squared and compared using a code loop discriminator. Based on the discriminator output a control signal is generated and locally generated C/A codes are adjusted to match the C/A code of the incoming signal. So called early-minus-late (E-L) envelope discriminator is used in DLL implementation. The locally generated prompt signal is used to despread the incoming signal.

The implemented PLL consists of DCO, carrier loop filter and a discriminator. PLL receives signal that is only modulated by navigation message. The DCO generates a carrier frequency based on the Doppler frequency computed during acquisition process. The generated signal is divided into in-phase and quadrature components. The input signal is correlated with I and Q signals. The outputs of the correlators are filtered and the phase is analyzed by a discriminator. The discriminator used in this implementation is arctangent discriminator that is insensitive to the bit phase transitions. Such PLL is similar to Costas loop. The output of the discriminator is used to generate a control signal to tune the frequency of the oscillator[2] so that the loop can continuously demodulate the incoming signal. We used the second order filter for PLL as described in [11]. The filter diagram is shown in Figure 5.10.

The time discrete description of the loop filter in the z-domain is given by:

[2]DCO

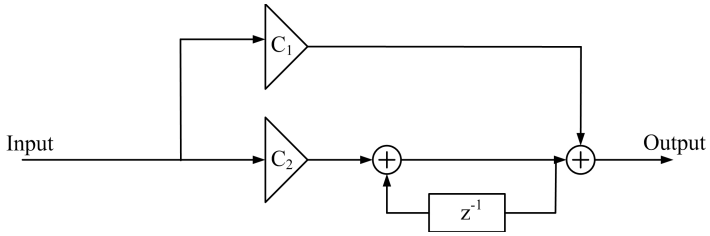


Figure 5.10. Carrier loop filter

$$F(z) = C_1 + \frac{C_2}{1 - z^{-1}} = \frac{(C_1 + C_2) - C_1 z^{-1}}{1 - z^{-1}} \tag{5.5}$$

where $C1$ and $C2$ are loop filter coefficients.

The coefficients $C1$ and $C2$ are given as:

$$C_1 = \frac{1}{k_0 k_1} \frac{8\zeta\omega_n t_s}{4 + 4\zeta\omega_n t_s + (\omega_n t_s)^2} \tag{5.6}$$

$$C_2 = \frac{1}{k_0 k_1} \frac{4(\omega_n t_s)^2}{4 + 4\zeta\omega_n t_s + (\omega_n t_s)^2} \tag{5.7}$$

where

$k_0 k_1$	-	loop filter gain
ω_n	-	filter natural frequency
ζ	-	filter damping factor
t_s	-	filter update time

Natural filter frequency ω_n is given by:

$$\omega_n = \frac{2B}{\zeta + \frac{1}{4\zeta}} \tag{5.8}$$

where B is the carrier loop noise bandwidth.

The parameter values of the carrier loop filter damping factor, natural frequency and loop gain are chosen by try and error approach.

The change in natural frequency (which depends on the noise bandwidth) and loop gain has effects on success or failure of the tracking, since these parameters are related with the incoming signal strength.

The damping ratio plays an important role in the performance of the tracking loop [21]. When system is underdamped ($0 < \zeta < 1$) the system response is rapid and will overshoot the desired state and will oscillate before settling down to the desired state. If the system is overdamped ($\zeta > 1$), the system response will be slow in achieving the desired state but will do so without any oscillation. The optimal response in the system is achieved when $\zeta = 0.707$ (system is critically damped). Thus this was the choice used for both carrier and code tracking loops in the receiver, but this is not the only solution.

The selection of the natural frequency of the loop is a compromise. A relatively small natural frequency will provide excellent noise performance but will be unable to track dynamics induced on the signal. A relatively large natural frequency will be able to track signal dynamics but will have poor noise performance.

The filter loop gain is chosen to be 400π. The update time t_s is chosen to be 1ms. The loop noise bandwidth chosen is 250 Hz. It is not the usual one, however it worked for the purposes of this project.

Below pictures illustrate the output of the tracking loops. Figure 5.11 shows the tracking performance of a GPS satellite and Figure 5.12 demonstrates the tracking performance of an EGNOS satellite.

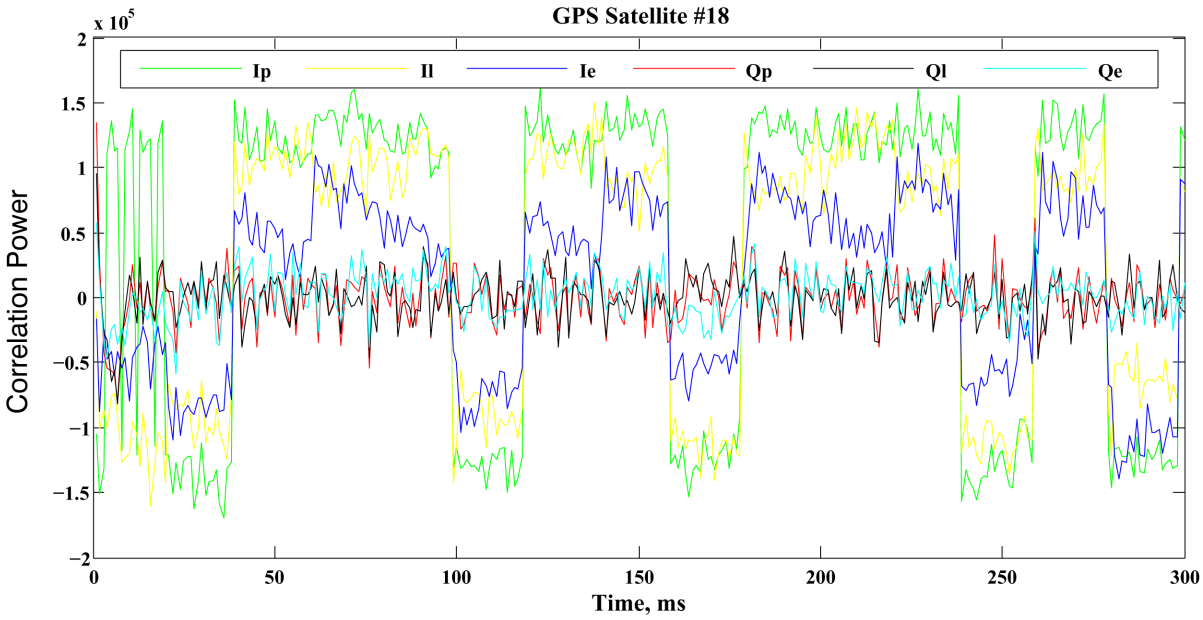


Figure 5.11. GPS tracking

What we wanted is to have the I component as large as possible while keeping the Q component as small as possible. Also prompt branches correlation should be higher than the one of

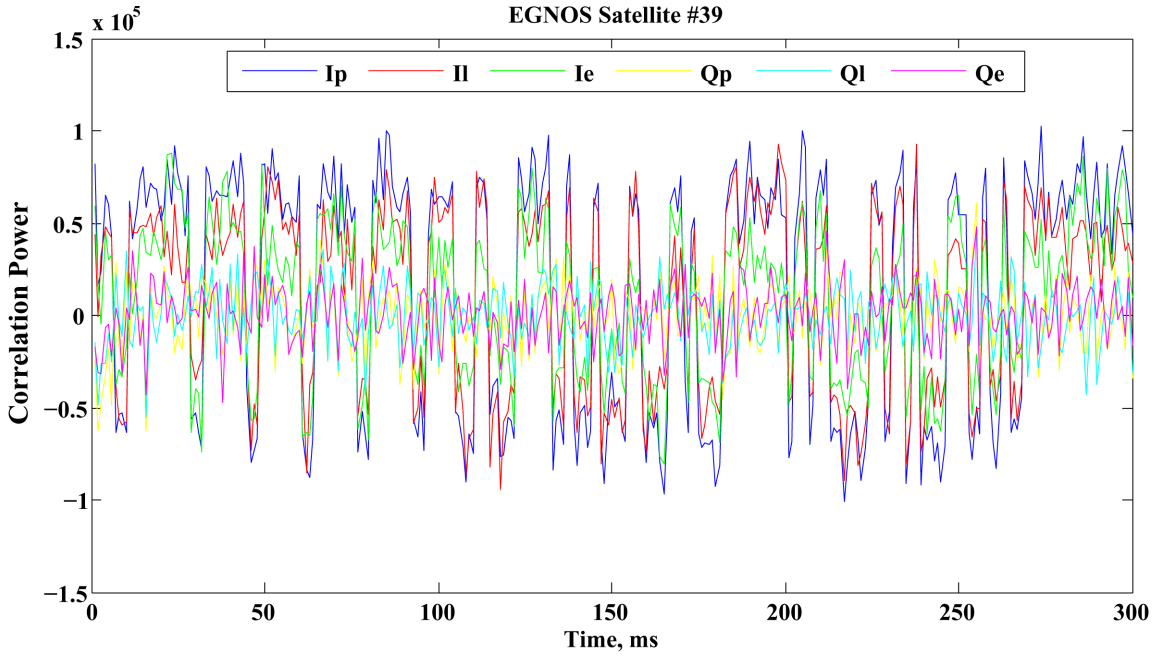


Figure 5.12. EGNOS tracking

early and late. Figures demonstrate this. The performance of GPS signal tracking was acceptable. However, as can be seen from Figure 5.12 EGNOS signal performance was not satisfactory enough. Some correlation power loss appeared. It was not enough time left to investigate the problem and to improve the performance. Due to this some tasks that were originally defined were not implemented. Implementation of bit synchronization and EGNOS data convolutional decoding is left for future refinement. This issues are shortly covered in the next three sections. Further the implementation of the EGNOS corrections application simulator is described.

5.3.1 Lock detectors

When the signal is being tracked, we need to know when it is tracking well and when it has lost track. Lock detectors are required to perform this function. The lock detector is a parameter that indicates the tracking status of the tracking loop. The receiver can make some status changes according its value. One way to implement a lock detector is by averaging the power over a certain period and comparing that to a present threshold. If the lock detector value is lower that the threshold, it means that the tracking loop has lost the lock. The receiver then, consequently, has to change the loop bandwidth, or even return to acquisition mode. This function has not been developed in the software receiver yet, and is an excellent candidate for development as a future perspective.

5.4 Bit Synchronization

The performance of EGNOS signal tracking was not satisfactory enough, thus bit synchronization algorithm was not implemented. In the case of GPS 20 ms constitutes one bit. If one of the samples in this sequence is wrong it is still possible to extract correct data bits since, when averaging the 20 samples with one or two wrong still will give the correct result. Therefore, the GPS bit synchronization techniques are robust. In the case of EGNOS only 2 ms constitutes a bit, thus an error in one of them will certainly lead to a wrong extracted bit, since when averaging two samples with one wrong certainly will give the wrong result. However, EGNOS data is convolutionally encoded with the purpose of transmission errors reduction. Thus, it still remains a probability, that obtained tracking results could lead to correct EGNOS data bits extraction from the signal. Although the EGNOS signal tracking performance certainly requires improvements.

5.5 Data Decoding

After data demodulation has been performed, the essential information in the signal can be extracted. In the case of GPS by parsing the received data bits, in the case of EGNOS performing additional data decoding, before parsing the data.

In this case the data recorded from real EGNOS receiver is used. This data is already decoded.

As was mentioned earlier in this report there is a standard MATLAB function for decoding convolutionally encoded data. This function implements a Viterbi's decoding scheme required to decode EGNOS data. The only way to test MATLAB function for Viterbi's decoding algorithm is to obtain EGNOS data from tracking loop. As mentioned above, since not the good performance of EGNOS signal tracking was obtained it was not possible to make use of this MATLAB function.

Only EGNOS data parsing was considered. This is described in the following sections.

5.6 Data Processing

5.6.1 Introduction

EGNOS and GPS data processing in this project is done in post-processing. The data is processed epoch by epoch and after each epoch the algorithm computes receiver position with EGNOS corrections. The algorithm structure is defined in Figure 5.13.

5.6.2 Data load

The data is stored into 2 data files: one for EGNOS and GPS data, and the second for earth geoid undulation parameters. The first one has the structure shown in Figure 5.14.

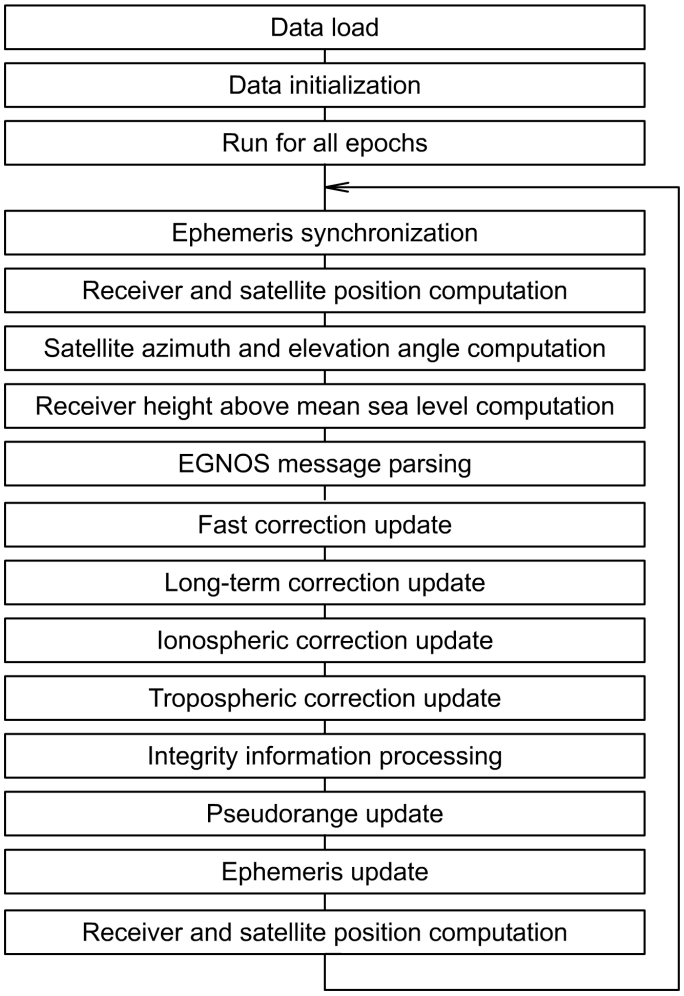


Figure 5.13. EGNOS data processing

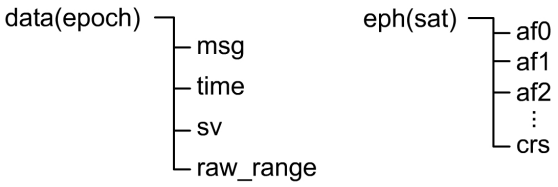


Figure 5.14. Data structure

Data structure component "data" stores EGNOS message (msg), GPS timestamp (time),

GPS satellite mask (sv) and raw pseudorange measurements for all SV's defined in the mask (Figure 5.14). Moreover, the "data" structure elements are arranged so, that it contains data set for each epoch. The second structure "eph" is used to supply ephemeris parameters from all satellites. Further, the data for each satellite is stored in array, where each satellite has a set of ephemeris for different update times (ephemeris parameters are listed in the Table E.1). Ephemeris data should be synchronized for current epoch to maintain up to date information. More detailed description is given in section 5.6.4.

The second data file is used for troposphere correction computations. The information from this file should be loaded into memory to compute receiver height above mean sea level. The data file used in this project is taken from National Geospatial Intelligence Agency [19]. Moreover, the file was processed using Matlab and stores earth geoid undulation values at specified points defined by the grid which has 0.25 degree spacing in latitude and longitude.

5.6.3 Data initialization

Data structures should be initialized before processing EGNOS and GPS information. This is needed in order to store whole data used in computations and data from previous computations. EGNOS system is very complex compared to the GPS and requires continuous collection of information which can be used later during ongoing processing. To simplify implementation the different types of data were proposed:

1. **Iono** structure stores ionospheric grid mask and corrections for all IODI values for four grid bands used by EGNOS.

2. **Eph** structure stores ephemeris parameters for all satellites.

3. **Sat** structure stores temporary and permanent parameters for all 37 GPS satellites defined by EGNOS. In this data structure each satellite will have the following categories of parameters:

    - **General**. Satellite mask bit, sequence number and IODP values.
    - **Fast correction**. Here the parameters for fast correction computation are stored. The most important are: data from the most recent fast correction message, from previous fast correction message, UDREI parameters, IODF parameters, fast correction degradation factor indicators and the final fast error correction value for the current epoch.
    - **Long-term correction**. Long-term corrections are less complex and only a few parameters have to be stored here: data from the most recent long-term correction message and the most recent long-term error correction value.
    - **Ionospheric correction**. Only the final ionospheric correction value is stored here.
    - **Tropospheric correction**. The final tropospheric correction value.

- **GPS data**. Includes: GPS satellite mask, raw pseudorange, corrected pseudorange, satellite azimuth angle and satellite elevation angle.

4. **Other parameters**. Other parameters used in computation usually are not satellite dependent and are used without any change. For example: time, receiver height above mean sea level, day-of-year, GPS week number and similar.

The above mentioned data structures should be initiated. The data structures "Iono" and "Sat" should be initiated only once, while other parameters should be initiated each time before the new epoch. The more detailed description of above-mentioned structures will be described in subsequent sections.

5.6.4 Ephemeris synchronization

Ephemeris is stored in three-dimensional structure and has data specified for different time periods. For proper assignment before each epoch, ephemeris should be synchronized to the current epoch. Moreover, synchronization is needed between broadcast ephemeris and EGNOS corrections in order to use the same ephemeris set for satellite position computation as it is used for corrections at EGNOS master control stations, otherwise, the errors will emerge. In this project an ephemeris is updated twice in one epoch:

1. **Ephemeris initialization.** In order to make pre-computations (refer section 5.6.5) the initial satellite and receiver positions have to be computed and thus, the initialization of ephemeris is needed. As the GPS constellation does not guarantee broadcast satellite ephemeris updates at regular predictable intervals, it should be checked constantly. According to [22] satellite ephemerides are broadcast every two hours earlier to the epoch for which they have been calculated. In this project ephemerides will be set valid if the current epoch is between two hours before the time of ephemeris (t_{oe}) and the t_{oe}. As the parameter t_{oe} is expressed in GPS time-of-week and there are possible crossovers at the end of week, the GPS time equivalent will be computed for t_{oe} as well as for current epoch:

$$\begin{aligned} GPStime &= week \cdot seconds_{week} + time \\ EPHtime &= wn \cdot seconds_{week} + t_{oe} \end{aligned}$$

where

$GPStime$	-	current epoch time expressed as GPS time in seconds
$EPHtime$	-	t_{oe} parameter expressed as GPS time in seconds
$week$	-	GPS week number for current epoch
wn	-	GPS week number from broadcast ephemeris
$seconds_{week}$	-	seconds in one week (604800)
$time$	-	current epoch time expressed as time-of-week in seconds
t_{oe}	-	time of ephemeris taken from broadcast ephemeris

Then, the time interval for certain ephemeris to be valid is computed as:

$$interval = [EPHtime - 2 \cdot seconds_{hour}; EPHtime]$$

where $seconds_{hour}$ is seconds in 1 hour and equal to 3600.

2. **Ephemeris synchronization with EGNOS.** This step is performed in order to synchronize EGNOS long-term corrections with broadcast ephemeris. Long-term corrections uses parameter IODE (issue of data ephemeris) which identifies broadcast ephemeris used in error computations at EGNOS MCC. This synchronization step is needed because the update of ephemeris in GPS satellites differs from update time used for correction computations. After GPS ephemeris update EGNOS system will continue to provide corrections for the old ephemeris set.

   Ephemeris is synchronized after parsing EGNOS messages and processing EGNOS corrections. It is achieved only if there are long-term corrections received for particular satellite, otherwise, the ephemeris is used as it was initially set by the first initialization procedure.

5.6.5 Pre-computations

Before decoding EGNOS messages some computations should be performed. The proceedings are listed below which is shown in Figure 5.15.

1. **Satellite position computation**. Satellite position is needed for further computations such as receiver position and satellite elevation angle. In order to perform this step one should supply algorithm with current time, valid ephemeris data and pseudoranges for all satellites in view. The algorithm is described in section E.1.

2. **User position computation**. After satellite coordinates are known the user position can be computed (refer section E.2). It is needed for:

   - Troposphere error computation: To compute troposphere error the geodetic user position (longitude, latitude, altitude) is needed. However, the receiver position obtained here is in ECEF coordinate frame and has to be transformed as described below in this section.

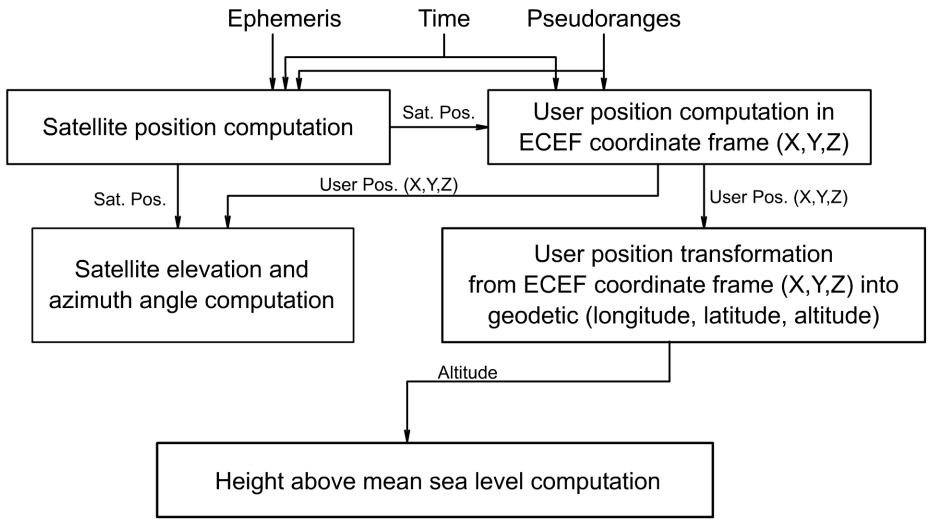


Figure 5.15. Pre-computations

- Satellite elevation angle computation: The satellite elevation angle is used in many algorithms and the user position is necessary to compute it.

3. **Satellite elevation and azimuth angle computation**. This procedure is one of the most important pre-computation procedures needed for ionosphere and troposphere error computations. Both error sources are related to the distance between satellite and receiver including the atmosphere. The satellite elevation angle is used to compute that distance and to convert ionosphere and troposphere error corrections from vertical delay estimation into the slant delay estimation by using obliquity factors (refer formulas 4.20 page 50 and 4.28 page 52). Moreover, the satellite elevation and azimuth angles are used to compute ionospheric pierce point (refer formulas 4.14 page 48 and 4.15 page 48).

4. **User position transformation**. Many procedures require user position to be expressed in different coordinate frame than in user positioning algorithms and, therefore, it should be transformed. This algorithm converts the user position from ECEF (X, Y, Z) into geodetic coordinate frame (longitude, latitude, altitude) (the source code is provided in section A.5), latter is used in various algorithms:

   - Ionospheric pierce point computation. In ionosphere error correction algorithm the IGPs are selected using the longitude and latitude of ionospheric pierce point, which is computed using user position longitude and latitude.
   - Tropospheric error correction. The meteorological parameters used in troposphere error correction are selected from the tables according to user latitude.

- Receiver height computation. The geodetic receiver position (latitude, longitude and altitude) is required to compute the height above mean sea level.

5. **Receiver height transformation**. This procedure applied to compute tropospheric delay correction. The height above mean sea level is computed from the user height above WGS-84 ellipsoid (altitude) and geoid undulation value. The transformation is done by using geoid undulation values that are stored in the file *egm96.mat*. The file is covering world-wide area by specifying the undulation values in the grid which is divided in 721 rows (180 degrees of latitude at 15 minute spacing) and 1441 columns (360 degrees of longitude at 15 minute spacing). As the undulation values are not given for all the areas precisely and has 15 minute spacing in longitude and latitude one should use its own coordinates (longitude and latitude) to interpolate the correct value. The interpolation procedure is linear and done in the following way:

   - Select four nearest grid points (Figure 5.16)

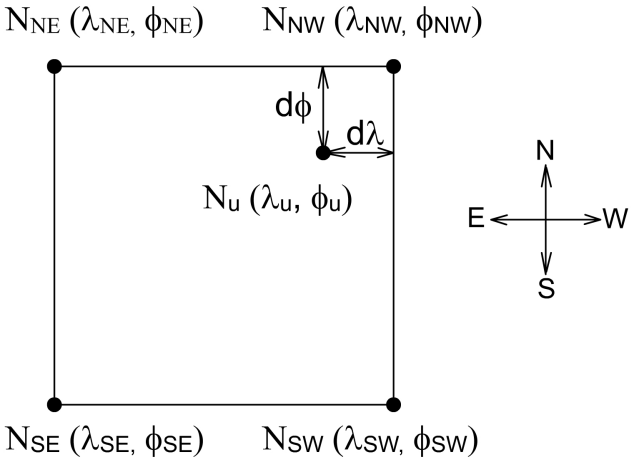


Figure 5.16. Geoid undulation value (N) interpolation

   - Compute the offset in latitude and longitude (d_λ and d_ϕ) from the selected northwest (NW) grid point.

$$\begin{aligned} d_\lambda &= \lambda_{NW} - \lambda_u \\ d_\phi &= \phi_{NW} - \phi_u \end{aligned}$$

   - Compute the longitude scale factor in upper and lower ($scale_{up}$, $scale_{down}$) part of the square.

$$\begin{aligned} scale_{up} &= N_{NE} - N_{NW} \\ scale_{down} &= N_{SE} - N_{SW} \end{aligned}$$

- According to the user position compute the undulation values for upper and lower parts of the square (N_{up}, N_{down}).

$$\begin{aligned} N_{up} &= N_{NW} + d_\lambda \cdot scale_{up} \\ N_{down} &= N_{SW} + d_\lambda \cdot scale_{down} \end{aligned}$$

- Compute the final undulation value (N_u) from the formula:

$$N_u = N_{up} + d_\phi \cdot (N_{down} - N_{up})$$

5.6.6 EGNOS message parsing

EGNOS messages are coming every second with different message type ID. Moreover, different type of messages have different update intervals which are precisely controlled by the transmitting part (MCC). Messages with shorter update intervals are kept more important and contain rapidly changing data such as integrity information (message type 0 and 6) of fast error corrections (message types 2-5 and 24). Messages with longer update intervals are less important and provide information which does not change rapidly. This include messages for long-term error corrections (message type 24-25), ionosphere error corrections (message type 18 and 26), PRN mask (message type 1) and others (Figure 5.17 shows the message distribution in 2 hour period). However, there are some messages which are not important and can be ignored depending on application. The target of this project is to correct GPS receiver errors, thus some of the messages will not be used. Used message types are: 0, 1-4, 6-7, 18 and 24-26. Before parsing messages two procedures are performed:

1. **Message synchronization**. Each message contains 8 bit preamble which is used to check message synchronization. If there were no errors during message decoding the preamble should match one of three predefined 8 bit words (01010011, 10011010 or 11000110) that will cycle from one to the other. If the sequence of these words is broken (the three successive messages should contain a 24 bit word consisting of three 8 bit words), or if the preamble does not match - the message will not be used in computations to avoid misleading information.

2. **Message parity check**. This check will find any random errors, the probability of undetected error $\leq 5.96 \cdot 10^{-8}$ as described in [6]. If the error is found the message will not be parsed and used in error correction computation.

In order to parse the received messages we check message type ID and use special parsing function for particular message. The parsing function converts the binary message content into information, which is used for error correction (parsing function's example is provided in section A.4).

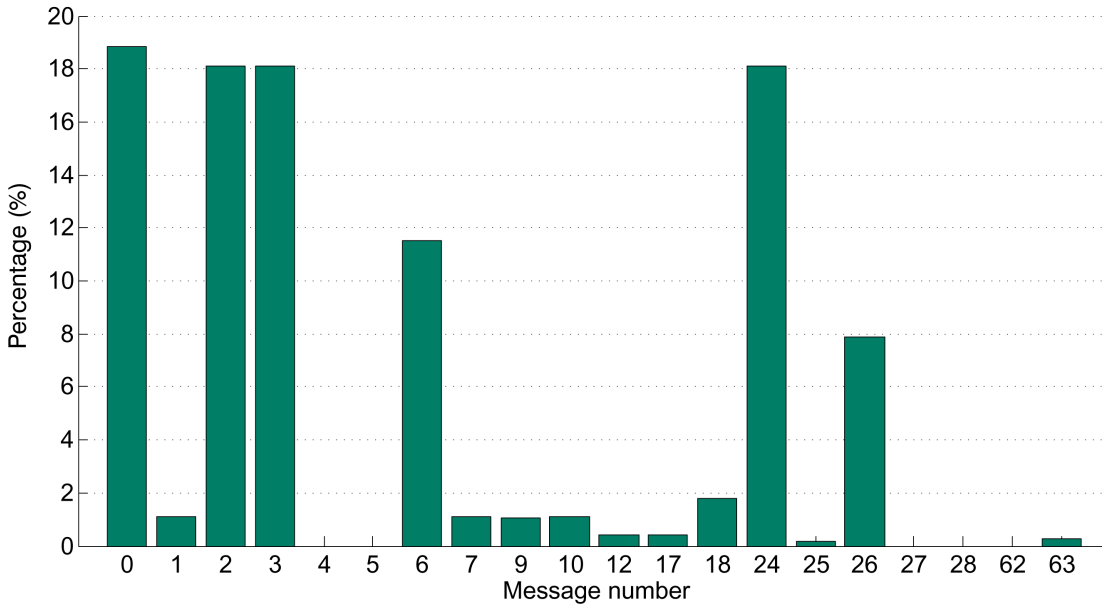


Figure 5.17. EGNOS message frequency (2 hour data)

5.6.7 Fast corrections

The overall fast error correction consists of PRC and RRC value combination. PRC value is directly received with fast correction messages (type 2-5 and 24) and the RRC is computed. The RRC value is used only if the fast correction message is not received and if the time-out conditions are met (refer section 3 page 43). The procedure for computation of PRC and RRC values is straightforward process and is provided in formulas 4.9 page 42 and 4.10 page 43. The error correction example is shown in Figure 5.18, where PRC values are received at epochs 2, 6, 11, 16, 21 and the RRC is computed and applied between these epochs. The Matlab code for fast error computation is provided below:

```
if ((sat(i).mask_bit == 1) && (sat(i).check_fast == 0) &&
    (sat(i).t_of ~= 0))
   % timeout conditions for RRC
   if (((sat(i).t_of - sat(i).t_of_prev) > sat(i).I_fc) && (sat(i).I_fc ~= 0))
      RRC = 0;
   elseif ((rec.time - sat(i).t_of - 1) > 8 * (sat(i).t_of - sat(i).t_of_prev))
      RRC = 0;
   elseif (sat(i).a_i == 0)
      RRC = 0;
   elseif  (sat(i).resetRRC == 1)
```

```
      RRC = 0;
   else
      RRC = (sat(i).PRC_cur - sat(i).PRC_prev) / (sat(i).t_of - sat(i).t_of_prev);
   end

   % fast error correction
   sat(i).fast_pseudo = sat(i).PRC_cur + RRC * (rec.time - sat(i).t_of);

% next time satellite will be updated using RRC
elseif (sat(i).check_fast == 1)
   sat(i).check_fast = 0;
end
```

The `sat(i).check_fast` parameter is set to 1 when the message with fast correction arrives for particular satellite. The RRC time-out conditions are computed if there are no fast correction messages received. Moreover, if we receive the message with corrections, RRC will cancel by default, because `(rec.time - sat(i).t_of)` will be equal to zero.

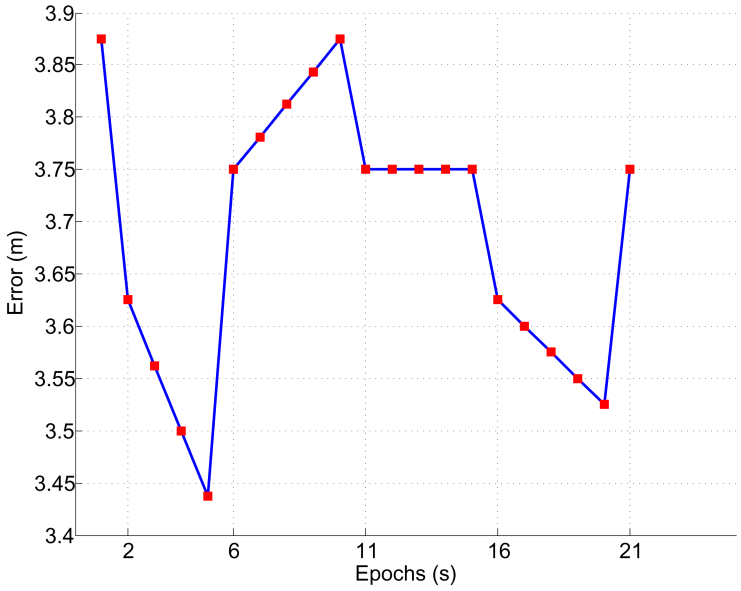


Figure 5.18. Fast error correction: PRC and RRC

The RRC correction was introduced in WAAS/EGNOS system to improve the performance with SA turned on and that was a great improvement to the overall performance. When the SA is turned off the RRC correction does not help much, as in former case. As we have seen in Figure 5.18 very often it can calculate wrong error prediction (e.g. epochs 3, 4 and 5). This

also can be seen in Figure 4.9 page 42. The comparison of fast error correction with RRC and without RRC is shown in Figure 5.19. To improve performance the RRC was analyzed more precisely and some tests where calculated to show the RRC influence on the output of the system (refer section 6.2).

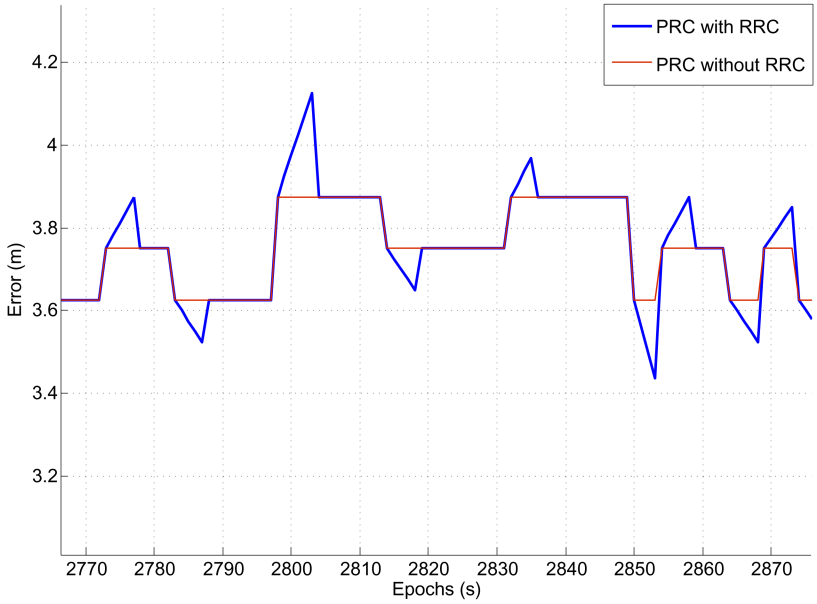


Figure 5.19. PRC with and without RRC correction

The example of fast error correction for about 2 hours is shown in Figure 5.20. As it is observed by experiment that the fast correction is usually within ±10 meter limit when SA is turned off.

User receives the UDREI values with fast corrections, which can show the general quality of corrections used (more information is provided in section 5.6.11 page 92). Moreover, the IODF values can identify the errors in fast error message transmission. Every time when the fast correction message is received for particular satellite, it will have different IODF values to identify continuous float (the IODF values will cycle: 0,1,2,0,1...). If the float is broken, user can identify transmission error and provide means to correct any uncertainty related to this situation (Figure 5.21 shows the IODF value change from epoch to epoch). However, the IODF value check is not implemented in this project.

5.6.8 Long-term corrections

The long-term correction is the easiest correction to implement. There are two corrections applied here: the satellite clock error correction and the satellite position error correction. The

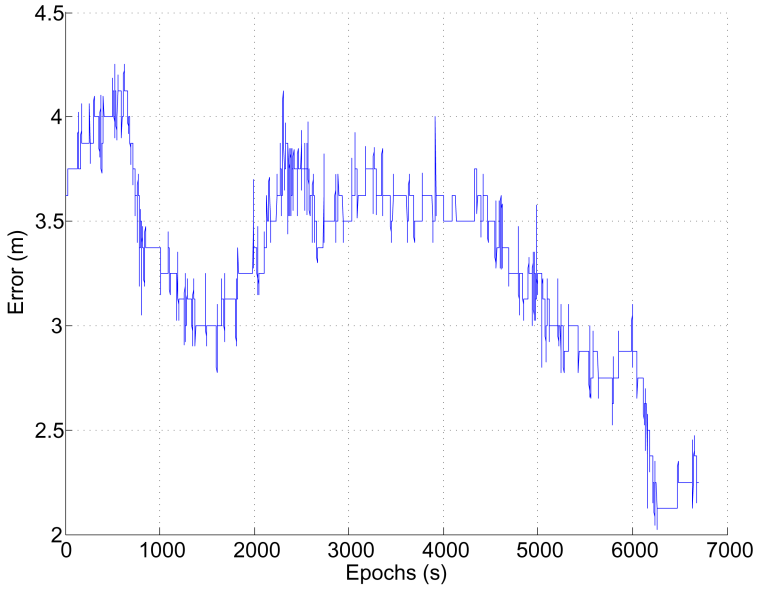


Figure 5.20. Fast error correction

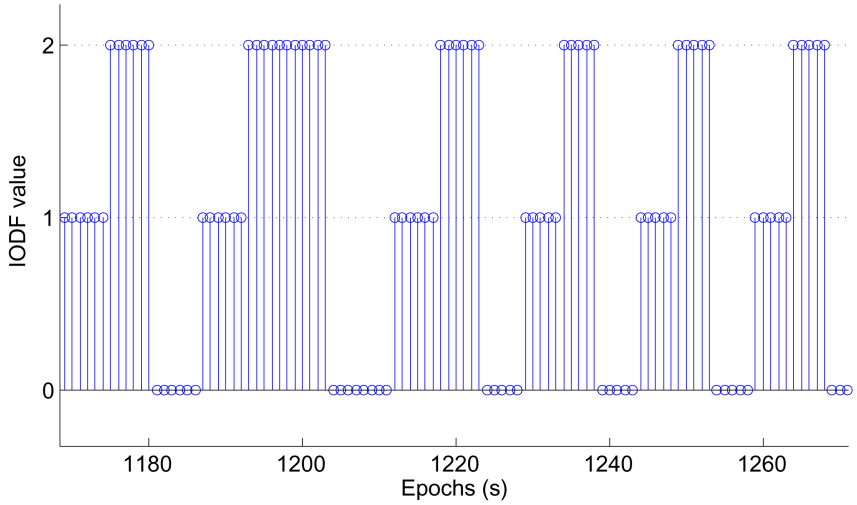


Figure 5.21. IODF

values of these corrections are used directly from the message type 24 or 25 (message type 25 appears rarely and the long-term corrections are received by message type 24, usually (Figure 5.17 page 79)). There are two ways to compute long-term correction:

1. Long-term correction with drift values. In this case long-term correction message will contain parameter $Velocity\ Code$ set to 1 and the satellite clock and position corrections will be computed using drift values, however, it is very rare and in practice we have never experienced it.

2. Long-term correction without drift values. In this case long-term correction message will contain parameter $Velocity\ Code$ set to 0 and the satellite clock and position corrections will be computed using drift values set to zero (0).

In both cases the long-term error correction will be computed using the same algorithm. The MATLAB code is provided below.

```
if sat(i).IssueOfData == mod(ephx(i).aode,256) &&
   sat(i).mask_bit == 1
     % Convert Time-of-Week to Time-of-Day
     timex = mod(rec.time,86400);
     % position error correction
     sat(i).slow_sat    = sat(i).delta +  (timex - sat(i).t_0) * sat(i).delta_r;
     % clock time error estimate
     sat(i).slow_sat1 = sat(i).af_0 + sat(i).af_1 * (timex - sat(i).t_0);
else
     sat(i).slow_sat1    =  0;
     sat(i).slow_sat     =  [0.0, 0.0, 0.0];
end
```

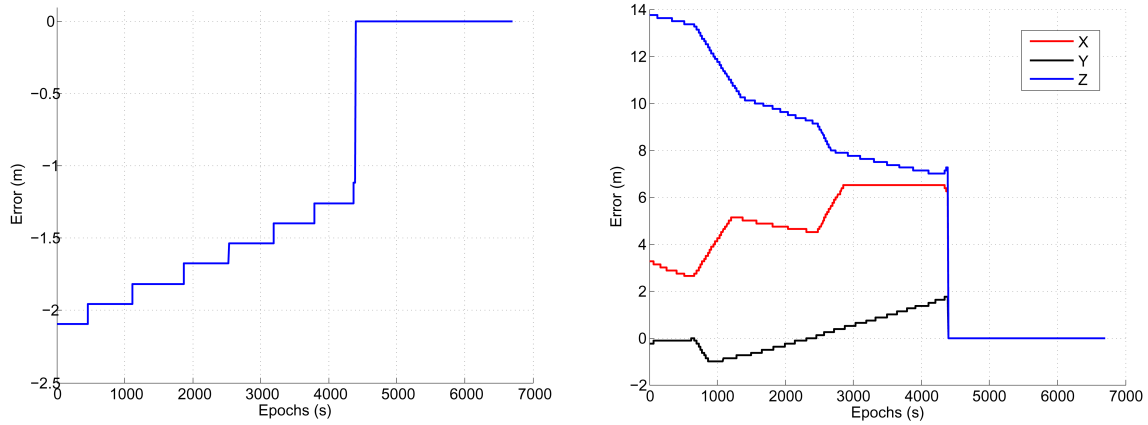
The implementation of long-term correction is solved in three steps, as meintioned in the code above:

1. Ephemeris check. The broadcast ephemeris should match the $IssueOfData$ parameter provided in long-term correction message.

2. Time conversion. The receiver time is corrected to represent time-of-day instead of time-of-week.

3. Correction calculation. Formulas 4.11 page 44 and 4.13 page 45 re used to compute the error correction.

The example of long-term corrections is shown in Figure 5.22. Figure 5.22(a) shows satellite clock error correction multiplied by speed of light and Figure 5.22(b) shows satellite position error correction.



(a) satellite clock error

(b) satellite position errors

Figure 5.22. Long-term error corrections

5.6.9 Ionospheric corrections

The ionosphere error correction is one of the most difficult error sources to implement. The data provided for this correction contains arrays of data which has to be stored permanently. The first step to obtain ionosphere error correction is to download the ionospheric mask. Then the corrections of each point in this mask will be collected and referenced. One should notice that in order to apply corrections the user should wait for corrections, which are covering his/her area. The structure for storing ionosphere grid mask and IGP vertical delay estimates, the following structure will be used:

```
%    iono.IODI_0.mask   - IGP mask for certain IODI(msg18)
%    iono.IODI_0.corr1  - IGP Vertical Delay Estimate (msg26)
%    iono.IODI_0.corr2  - GIVEI (msg26)
%    iono.IODI_0.corr3  - GIVE_i (msg26)
%    iono.IODI_0.corr4  - sigma_(i,GIVE) ^ 2 (msg26)

% the structure contains all IODI values which can vary from 0 to 3.
% Each IODI has mask, corr1, corr2, corr3 and corr4 (see below)
% iono.IODI_0. ...
% iono.IODI_1. ...
% iono.IODI_2. ...
% iono.IODI_3.mask
%     -//-   .corr1
%     -//-   .corr2
%     -//-   .corr3
```

```
%     -//-   .corr4

gg = 'IODI_0'; for i = 0:3
    iono.([gg(1:5),(gg(6)+i)]).mask  = zeros(35,32);
    iono.([gg(1:5),(gg(6)+i)]).corr1 = zeros(35,32);
    iono.([gg(1:5),(gg(6)+i)]).corr2 = zeros(35,32);
    iono.([gg(1:5),(gg(6)+i)]).corr3 = zeros(35,32);
    iono.([gg(1:5),(gg(6)+i)]).corr4 = zeros(35,32);
end
```

One should notice that ionosphere mask will have IODI value which is the most important parameter in identifying vertical delays. There can be situation when the IODI will change and the user will have mask with different IODI value which in normal condition will reset ionospheric computations. This can issue some jumps in position computation and also, the data can degrade. Moreover, when the IODI value changes in the mask some ionosphere corrections may still arrive using the old IODI values. To avoid this, the IODI parameter should be changed smoothly by saving old data associated with old IODI value. Such situation forces us to store all the incoming corrections by having separate structure elements for different IODI values as it is shown above.

As one can see from the MATLAB code above, the structure initiation is done by setting all values to zero. Moreover, the data is stored in arrays in size of 35 rows (latitude spaced by 5 degrees from N85 to S85) and 32 columns (longitudes spaced by 5 degrees from W60 to E100). Such array will fully cover four bands monitored by EGNOS. However, after performing some monitoring of data provided via ionosphere grid mask and vertical delay estimates we saw that the system will cover area, which is much smaller (Figure 5.23 shows monitored grid points in white, and available vertical delay estimates in red).

The next step in ionosphere error correction is to find the ionopheric pierce point. The procedure is straightforward as it was defined on page 48. The function for computing ionospheric pierce point is provided in section A.7. The computation of pierce point begins by running the following code:

```
 if mask_bit_GPS(i) == 1
     corr  = iono.([gg(1:5),(gg(6)+ IODI)]).corr1;
     corr1 = iono.([gg(1:5),(gg(6)+ IODI)]).corr4;
     mask  = iono.([gg(1:5),(gg(6)+ IODI)]).mask;
     zz    = iono_new(sat(i).satpos_raw(1), sat(i).satpos_raw(2),
             sat(i).satpos_raw(3), rec.posLLA(1), rec.posLLA(2),
             mask, corr, corr1, sat(i).sat_el, sat(i).sat_az);
     sat(i).iono_pseudo = zz.IC;
else
     sat(i).iono_pseudo = 0;
end
```

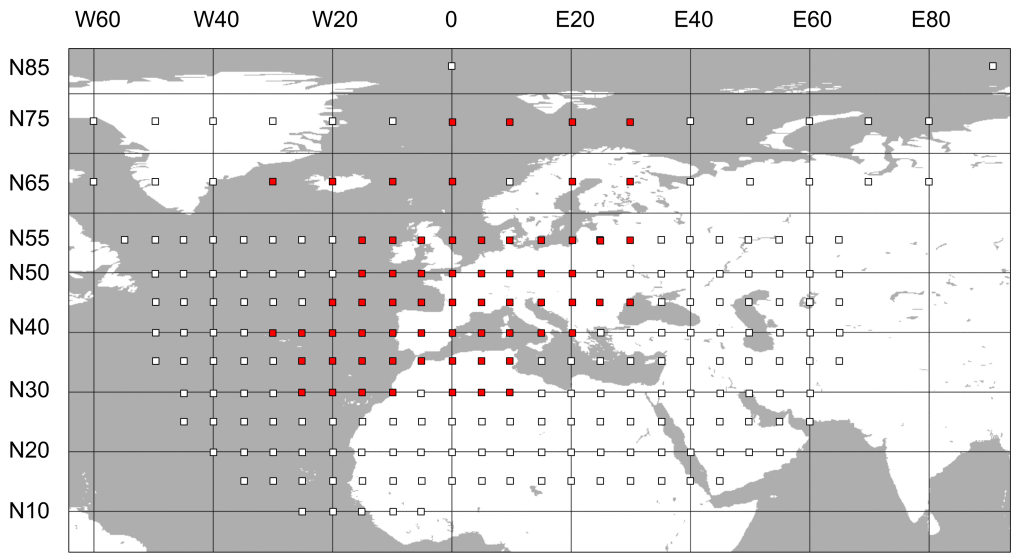


Figure 5.23. EGNOS ionospheric grid mask

The function `iono_new()` will compute ionospheric error (`zz.IC`). The listings of this function is provided in section A.6. The function `iono_new()` will start computations by finding the Ionosphere Pierce Point (IPP).

After the IPP is known the grid point selection is performed. This step requires selecting the ionospheric grid points around the computed IPP. The example of grid points surrounding the IPP and user position is shown in Figure 5.24. In order to interpolate vertical delay estimate value for computed IPP we will select 4x4 matrix of IGP's around it (selection will cover 10x10 degree square with IPP in the middle of 5x5 degree inner square as shown in the Figure 5.24). This allows us to use all possible combinations of IGP selection as defined in WAAS interface document [6] (algorithm is described in section D.2). The selection procedure is implemented in the function `iono_new()` (section A.6), which will take mask, vertical delay estimate and GIVEI arrays as inputs and will extract 4x4 matrices covering only the IPP area (in Figure 5.25 one can see the example of selected GIVEI and vertical delay estimate values). In figure the GIVEI value "NaN" means that the defined grid point is not monitored and can not be used in computation. Moreover, in order to provide reliable receiver position computation solution, the points with GIVEI values higher than 4 are not used in computation as well as values equal to zero (0 value means that there is no correction available at particular IGP).

Once the 4x4 surrounding matrices are extracted, the algorithm selects available 4 or 3 points for interpolation procedure (refer D.2). The procedure is quite complex, because there are 21 combinations available for selection of available IGP's for interpolation. The procedure should check all cases one by one until solution is reached or no points are selected. The selection

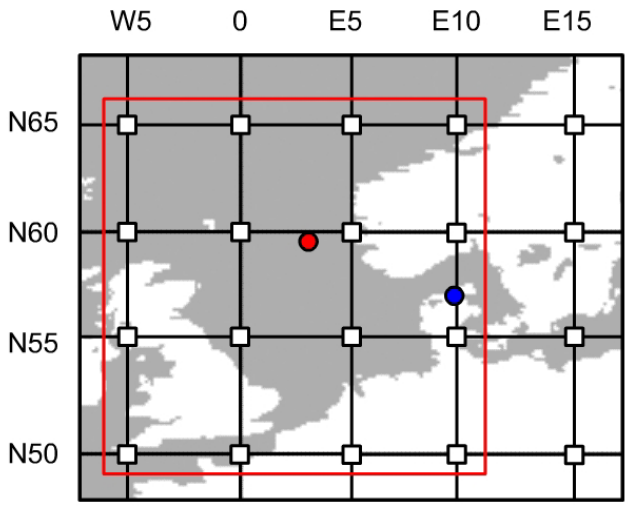


Figure 5.24. IGP selection. Blue dot - user position, red dot - pierce point

0	0.2994	0	NaN
0	0	0	0
0.2079	0.2079	0.2079	0.2079
0.2079	0.2079	0.2079	0.2079

0	0.875	0	63.75
0	0	0	0
0.375	0.375	0.375	0.375
0.5	0.5	0.625	0.625

Figure 5.25. Values of selected points: left - σ^2_{GIVE}, right - Vertical delay

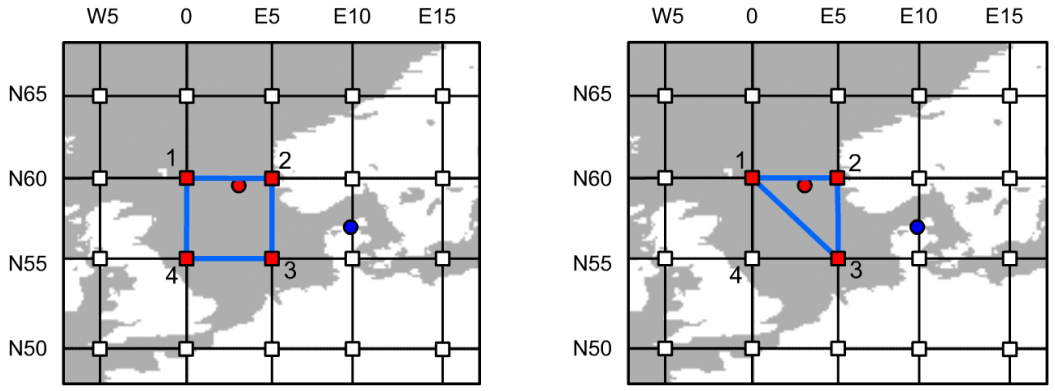
algorithm is listed in following steps:

1. 5x5 degree square selection. User should check if all four points defined in a square are available for selection (points with GIVEI values less than 4 and not equal to 0). The Figure 5.26(a) shows the checked square. If all points are available - these points are selected for interpolation, otherwise we move to the next level.

2. 5x5 degree triangle selection. If 3 of the 4 points from the previous section are available, then there are 4 triangles that can be used to get those 3 points for selection (Figure 5.26(b)). One requirement is that the IPP has to be inside the selected triangle and this state is checked by using function `triangle()` described in section A.8. If none of four cases were successful we move to the next level.

3. 10x10 degree square selection. There are 4 available squares around the IPP as shown in Figure 5.27(a). All the squares are checked as it was done using 5x5 degree square. If there are still no points selected we move to the next level.

4. 10x10 degree triangle selection. This is the last check we can make, we take each of the square from 10x10 degree matrix and check for four triangles inside that square. The procedure is the same as it was done with 5x5 degree triangle (Figure 5.27(b)). If there are no points available after this procedure, then the ionosphere correction can not be computed[3].



(a) 5x5 degree square

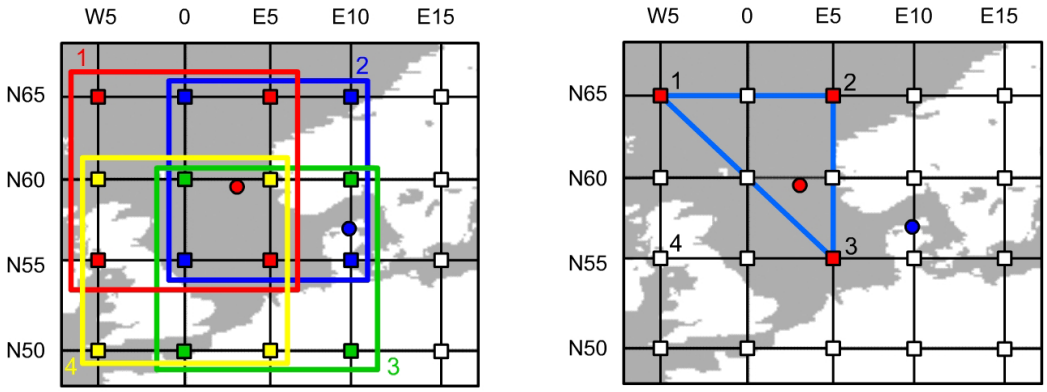
(b) 5x5 degree triangle

Figure 5.26. IGP selection

If there were any available points for ionosphere error computation the interpolation is performed using 3 or 4 selected IGP's (depends on the algorithm of point selection). Interpolation is implemented in the following way:

```
if shape == 4 && found == 1
    % Weighting function for 4 point interpolation
    W = [x_pp * y_pp, (1 - x_pp) * y_pp, (1 - x_pp) * (1 - y_pp), x_pp * (1 - y_pp)];
    a.corr   = sum(W .* corrx);
    a.UIVE   = sum(W .* corrx1);
elseif shape == 3 && found == 1
    % Weighting function for 3 point interpolation
    W = [y_pp, 1 - x_pp - y_pp, x_pp];
    a.corr   = sum(W .* corrx);
```

[3]There are more selection techniques defined for special areas such as south or north pole. In this project these methods are not implemented



(a) 10x10 degree square (b) 10x10 degree triangle

Figure 5.27. IGP selection

```
    a.UIVE   = sum(W .* corrx1);
else
    a.corr   = 0.0;
    a.UIVE   = 0.0;
end
```

The final output after interpolation will be the vertical delay estimate value for the IPP (`a.corr`). The ionosphere obliquity factor and ionosphere error correction are computed as (function `inon_new()`):

```
% Obliquity factor
F_pp = (1 - ((R_e * cos(el)) / (R_e+h_I))^2)^(-0.5);
% Ionospheric delay correction
a.IC   = - F_pp   * int.corr;
% sigma^2 UIRE
a.UIRE =   F_pp^2 * int.UIVE;
```

Obliquity factor is used here to transform ionosphere vertical delay estimate into the slant delay estimate. The function is directly related to the satellite elevation angle (Figure 5.28).

The final ionosphere error correction is shown in Figure 5.29.

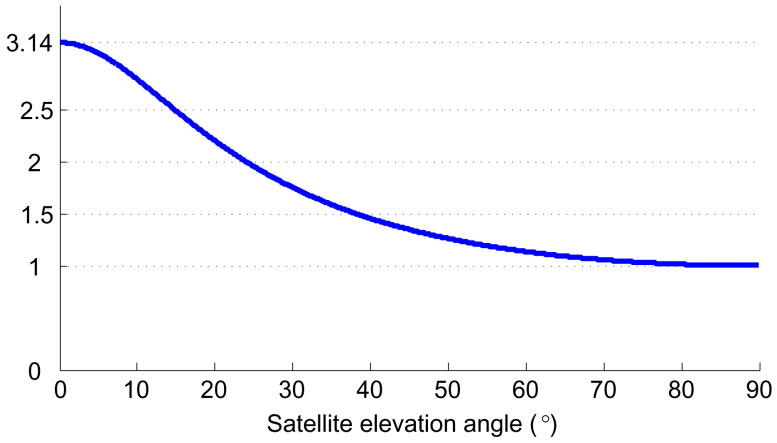


Figure 5.28. Ionosphere obliquity factor

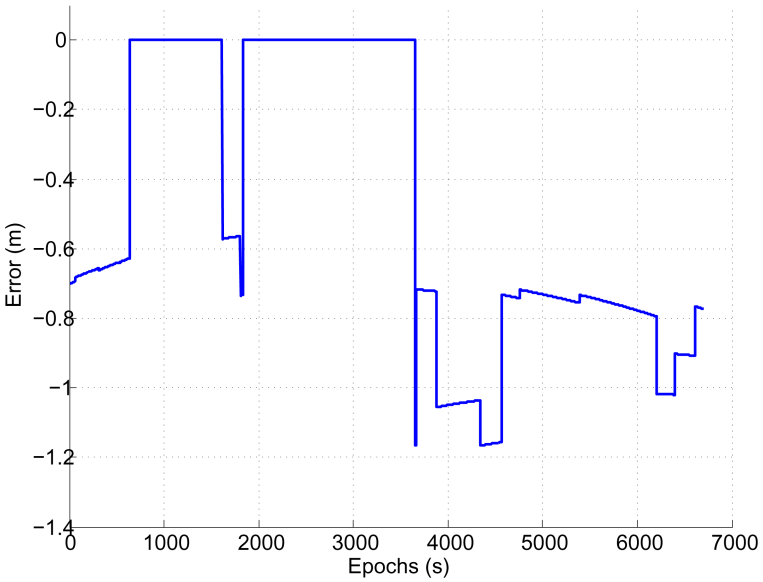


Figure 5.29. Ionosphere delay correction

5.6.10 Tropospheric corrections

To compute this error we call the function `tropo_new()`, which uses satellite elevation angle, receiver latitude, height and day-of-year parameters:

```
if mask_bit_GPS(i) == 1
```

```
      zz  = tropo_new(rec.day, rec.posLLA(1), sat(i).sat_el, rec.H);
      sat(i).tropo_pseudo = zz.corr;
else
      sat(i).tropo_pseudo = 0;
end
```

The function `tropo_new()` can be referenced in section A.9.

Troposphere error computation mainly depends on day-of-year and satellite elevation angle. Day-of-year changes troposphere error amplitude mainly because of meteorological parameter seasonal variations. The influence to final solution is shown in Figure 5.30. The satellite elevation angle is used to transform the troposphere vertical delay into slant delay. To realize this 2 obliquity factors are used: m_{hyd} and m_{wet}. The influence of satellite elevation angle to the obliquity factors and to the final error computation is shown in Figures 5.31(a) and 5.31(b).

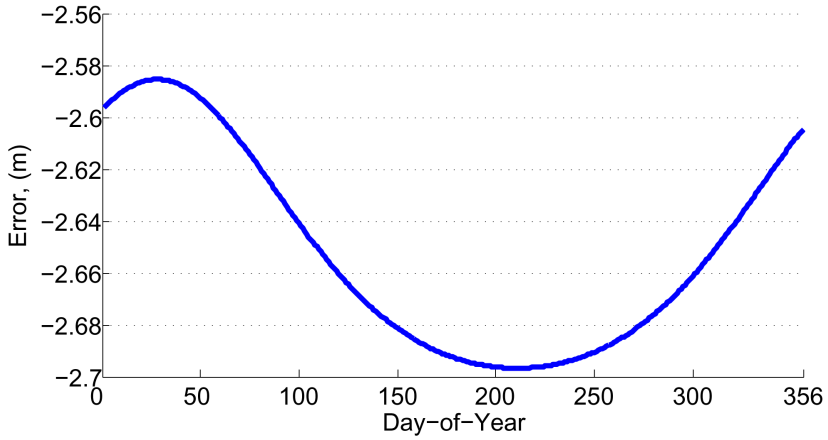
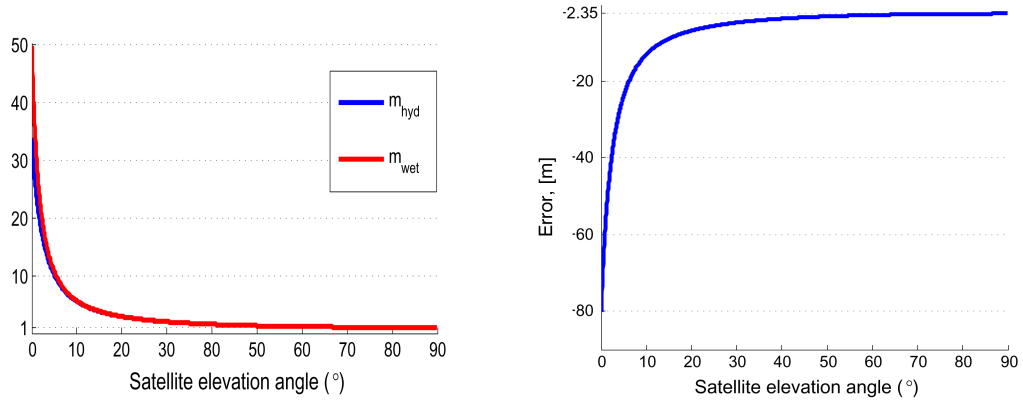


Figure 5.30. Troposphere delay error (elevation: 65°)

Finally, the Figure 5.32 shows the typical picture of the troposphere delay correction obtained from the algorithm.



(a) Obliquity factors (b) Troposphere delay error

Figure 5.31. Troposphere delay error

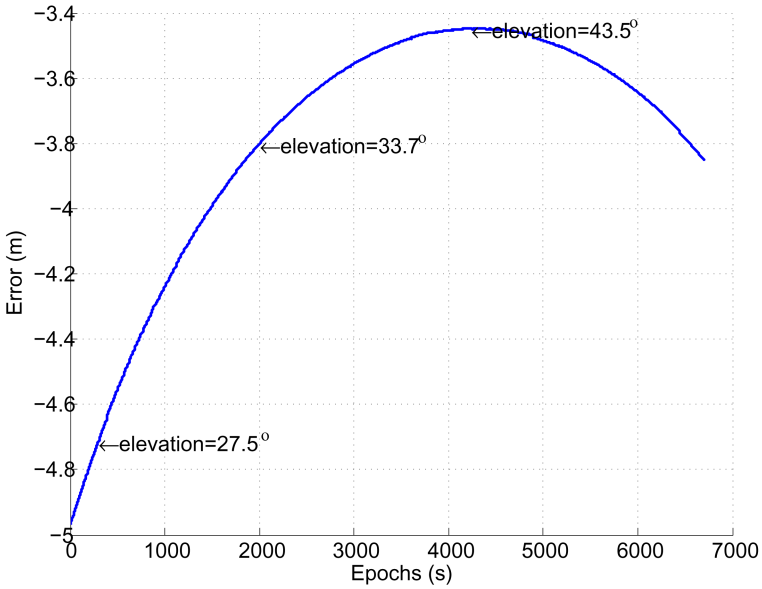


Figure 5.32. Troposphere delay correction

5.6.11 Quality control

Fast correction messages (2-5 and 24) and integrity information messages (6) contain integrity indicators in the form of User Differential Range Error (UDRE) estimates. These UDRE are an upper bound - in form of a standard deviation - on the residual error of the pseudorange after

the application of fast corrections, including the possibility that any messages are missed by the user [17]. The UDRE are used to compute protection levels and warning flags indicating that an individual PRN should not be used in the position solution. The code, which controls the fast correction usage according to the UDRE indicators (UDREI) is shown below:

```
for j = 1:37
  if sat(j).UDREI > 13 && mask_bit_GPS(j) == 1
      mask_bit_GPS(j) = 0;
  end
end
```

According to [6] fast correction should not be used if UDREI values are 14 ("not monitored") or 15 ("do not use"). The Figure 5.33 shows the UDREI values for particular satellite.

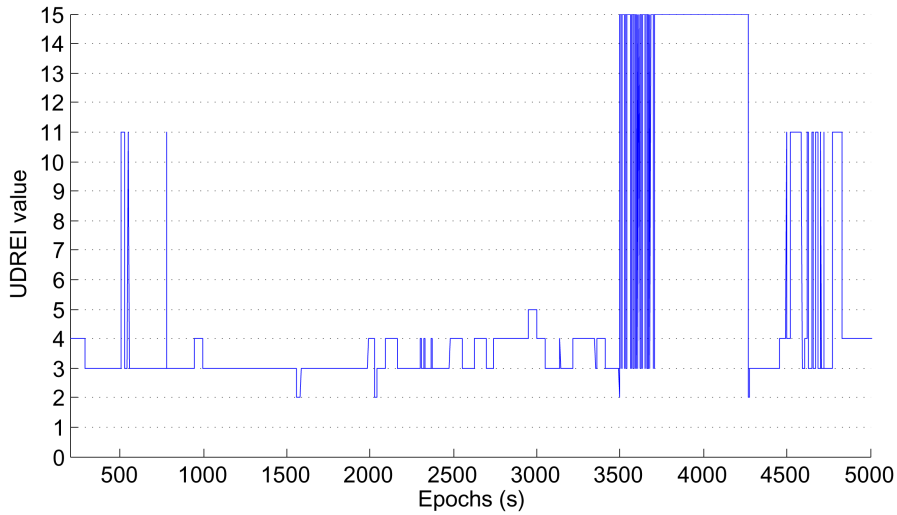


Figure 5.33. UDREI

Ionosphere correction is also controlled by quality indicators which is called the Grid Ionospheric Vertical Error Indicator (GIVEI). During selection of IGP's for vertical delay estimate interpolation, the GIVEI indicators control that there should not be any points selected with GIVEI values greater than 13. The code for IGP removal with GIVEI $>$ 13:

```
% remove corrections with sigma^2 GIVE > 30, - where GIVEI= 14-15
for gg = 1:4
  for gg1 = 1:4
    if corr1(gg,gg1) > 30 || isnan(corr1(gg,gg1)) == 1
      corr1(gg,gg1) = 0;
      corr(gg,gg1)  = 0;
```

```
    end
  end
end
```

Moreover, the IGP points are not selected if the ionosphere mask is not set to 1 for particular IGP.

5.6.12 Position correction

After processing data for an epoch the final step is to use computed corrections. There are two updates required:

1. Pseudorange update. Here we add troposphere, ionosphere, satellite clock and fast corrections (MATLAB code is provided below).

```
% initialization
pseudo_egnos = zeros(1,37); %
sat_corr     = zeros(37,3);%
for i = 1:37
  if mask_bit_GPS(i) == 1
    % pseudorange update
    pseudo_egnos(i) = sat(i).pseudo_raw  + sat(i).fast_pseudo/c + sat(i).iono_pseudo/c +
                          sat(i).tropo_pseudo/c + sat(i).slow_sat1;
    % satellite position update
    sat_corr(i,:)   = sat(i).slow_sat;
  else
    pseudo_egnos(i) = 0;
    sat_corr(i,:)   = [0; 0; 0];
  end
end
```

2. Satellite position update (MATLAB code above).

Finally, the new updated receiver position is calculated by calling the function `recpos2()`:

```
ff = recpos2(pseudo_egnos , rec.time, ephx, mask_bit_GPS,
     sat_corr); %
rec.posXYZ_EGNOS = ff.pos;
```

Where

`pseudo_egnos`	-	updated pseudorange
`rec.time`	-	receiver time
`ephx`	-	ephemeris for current epoch
`mask_bit_GPS`	-	satellite mask
`sat_corr`	-	satellite position correction

For results refer chapter 6.

5.7 Summary

The receiver architecture is designed. Some issues related to acquisition such as frequency refinement, threshold estimation and comparison between the correlators have been discussed. The data decoding and structures are elaborated. The data processing includes Data load, initialization, ephemeris synchronization, receiver and satellite position computation, computing satellite azimuth and elevation angle, receiver height, EGNOS message parsing, Fast correction update, long-term correction update, ionospheric and tropospheric update, Integrity information and pseudorange update etc. The method for these computations are illustrated with the help of necessary MATLAB code snippets.

Chapter 6

Tests

This chapter contains the tests for implemented parts of the software receiver. The first part involves the acquisition and tracking of GPS/EGNOS broadcast signals. The second part involves the results of combination of EGNOS decoded data with GPS data.

When the raw sampling data is collected, the antenna is fixed on top of the building, thus, only the static data is used. The study of error source like multipath or signal gaps is not main purpose of this project and the tests under these scenarios are not implemented.

6.1 Signal Acquisition and Tracking

Software approach main advantage is that it gives possibility to test many new algorithms and to modify existing ones easily. Performance of two different acquisition schemes that were implemented was verified against each other. The table shortly summarizing the comparison results is presented below.

Parameters	**Normal FFT-based acquisition**	**Modified averaging correlator acquisition**
Satellite	EGNOS 39	EGNOS 39
Coarse frequency	3563000	3563000
Code starting point	7614	7613
Processing time	63.481 s	18.146 s

Table 6.1. Acquisition comparison

From Table 6.1 it can be seen that the coarse frequency estimated by both methods algorithms was the same. Performance of modified averaging correlator did not degrade in this aspect when compared to standard FFT-based acquisition. The estimation of the code phase

(its starting point in data) provided by two methods was different. The difference of code phase estimations is 1 sample. All points FFT should provide and exact code phase within accuracy of one sample. This means that estimation of code starting point done by modified averaging correlator was 1 sample wrong. Modified averaging correlator acquisition was evidently faster than normal FFT based acquisition module. The timing was measured with the help of MATLAB functions *tic* and *toc*. No statistics on acquisition was investigated. The probability of signal detection and probability of false alarm was not computed. Approvement of satellites acquisition or non acquisition was done by investigating the acquisition plots. If a correlation peak was observed and the algorithm reported that the satellites were acquired, then it was known that the algorithm performed well, and did not give any missdetection. From this comparison it was concluded that acquisition algorithms performed well.

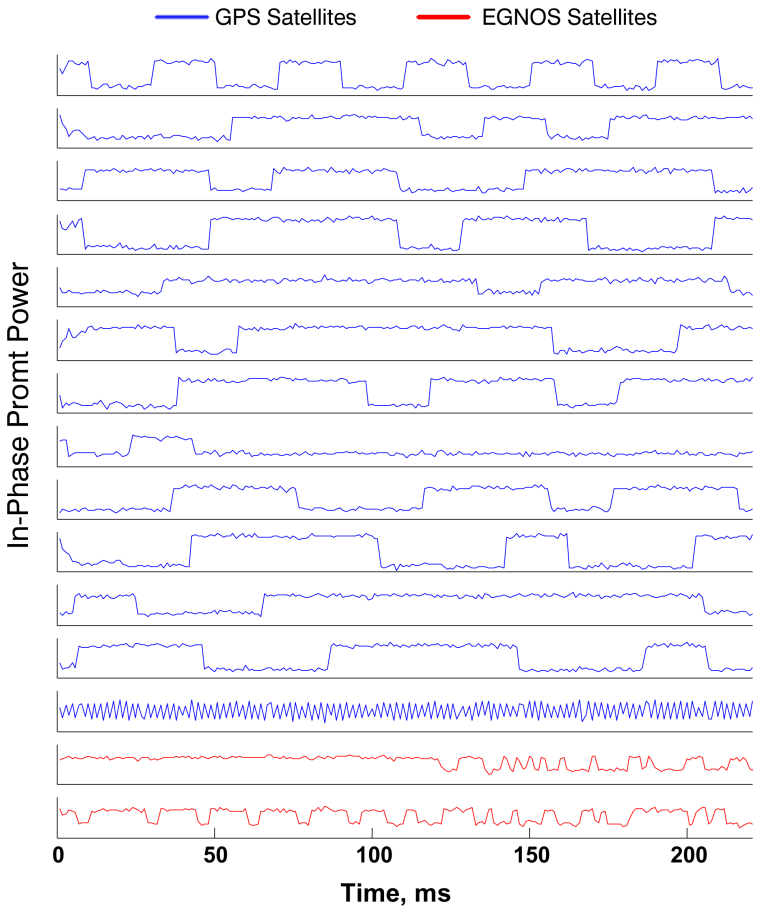


Figure 6.1. Receiver tracking performance

By testing acquisition of different satellites it was found that the code phase estimation by modified averaging correlator can be maximally 4 samples off from the true one estimated by all points FFT correlator. When testing the tracking loops performance with the parameters estimated by different acquisition algorithms it was found that wrong code phase estimation does not degrade the performance of tracking loops if a SNR is high enough. If SNR of the signal is low, then the satellite, which acquisition parameters were estimated wrong, can not be tracked.

Receiver could track almost all satellites that were acquired. The figure below (Figure 6.1) illustrates the tracking of 15 satellites. First 13 of these are GPS satellites and last two are EGNOS. One of the GPS satellites could not be tracked.

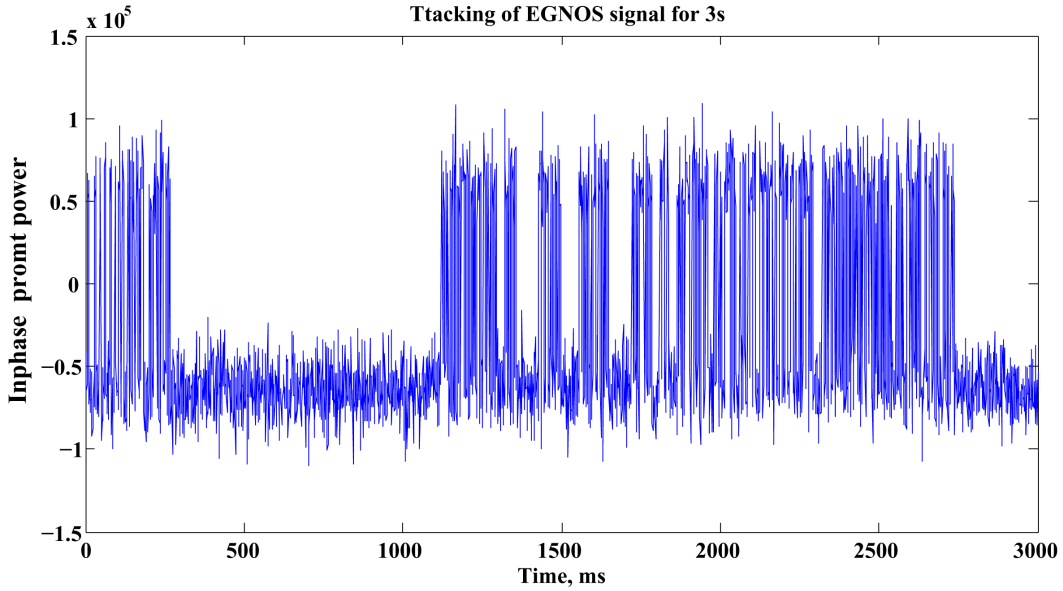


Figure 6.2. EGNOS signal tracking

The tracking was done on approximately 300 ms of data. Tracking of EGNOS satellites is not good enough. However, it is evident that the EGNOS satellites are tracked and no signal loss occurs. The tracking of EGNOS satellite on 3s of data is shown on Figure 6.2.

It is seen that the signal was tracked for all 3s. From this it could be concluded that EGNOS signal tracking could be improved with slight modifications to currently implemented algorithms. However, as well it is necessary to mention that it is unknown how just the hardware parameters changes would influence the tracking algorithms performance. It is possible that just usage of another RF front end with different sampling frequency will improve the EGNOS tracking performance and no any changes will be required to algorithms. These issues remain for future investigations.

6.2 EGNOS Corrections

In this part we will try to show the output of EGNOS system as comparison to the raw measurements obtained through GPS receiver.

Building the system our target was to get as best results as we could get. Our early tests with EGNOS system produced results out of scope and led us in many discussions and improvements which are not solved yet. However, only few technical details remain unsolved which we will try to showcase here.

First, we tried to investigate all the error corrections used by EGNOS separately. The Figures 6.3(a), 6.3(b), 6.4(a) and 6.4(b) show how different corrections improve or downgrade the final user position. The figures are organized so, that updated user positions can be compared to the raw GPS measurements computed without EGNOS corrections (blue dots) and to the GPS measurements obtained through professional C/A Thales DG14 receiver[1] (pink dots). Moreover, the tests were done in fixed position with antenna mounted at the very well known position which is referred as reference position with coordinates equal to (0,0). The position shown here is expressed in two-dimensional plane using ED50 datum transformation. X axis represents Easting and Y axis Northing. The tests are done using 2 hours data with an epoch time of 1 second.

The pseudorange errors for each EGNOS error correction (except satellite position correction) are shown in Figure 6.5. In this figure you can see the influence of different error sources to the raw pseudorange (data is provided only for one satellite). Moreover, you can see the total error correction as the thick red line.

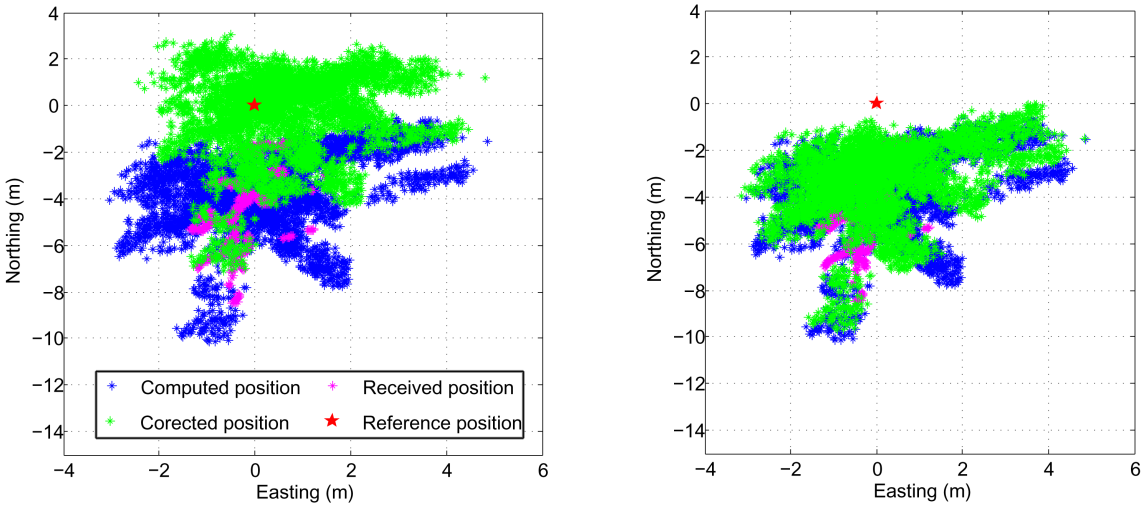
The final position correction is done by adding all error corrections. The results of this is shown in Figure 6.6 with characteristics given in Table 6.2.

In Figure 6.6 one can see that corrected position is better than computed position, unfortunately, the accuracy is still poor. According to the statistical data in Table 6.2 the accuracy and precision of the output is close to the position computed in Thales GPS receiver and much better than position computed using raw GPS data.

The results were not good and made us to look for better solutions or improvements which can eliminate the uncertainties in results. There were some improvements which arose absolutely accidentally while analyzing data. We saw that without adding long-term corrections into account the position solution improves dramatically. The output you can see in Figure 6.7 and characteristics comparison with previously obtained results in Table 6.3.

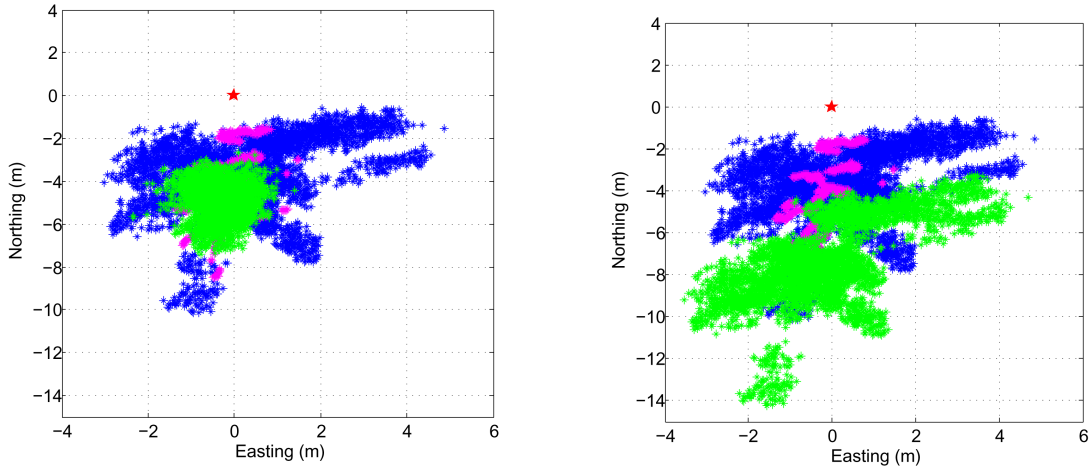
Taking the long-term corrections out of position computation we obtained results, which are within the accuracy limits defined by EGNOS. However, the EGNOS system defines that long-term correction should be used as often as it is available, thus, it is obvious that our algorithms still need some debugging and improvements. Fortunately, that also means that the solution can be improved even more.

[1]The Thales receiver documentation specifies that receiver stand-alone (not DGPS) position accuracy is 2 meters Circular Error Probable (CEP).



(a) Fast (b) Ionosphere

Figure 6.3. EGNOS correction influence to receiver position (1)



(a) Troposphere (b) Long-term

Figure 6.4. EGNOS correction influence to receiver position (2)

According to the [6], the corrections should be applied using the received UDREI values, which should show which satellites can be used in computation and how good are the corrections. In this test we were removing from computations satellites, which have been selected as not usable (UDREI values 14 ("not monitored") and 15 ("do not use")). The results are shown in

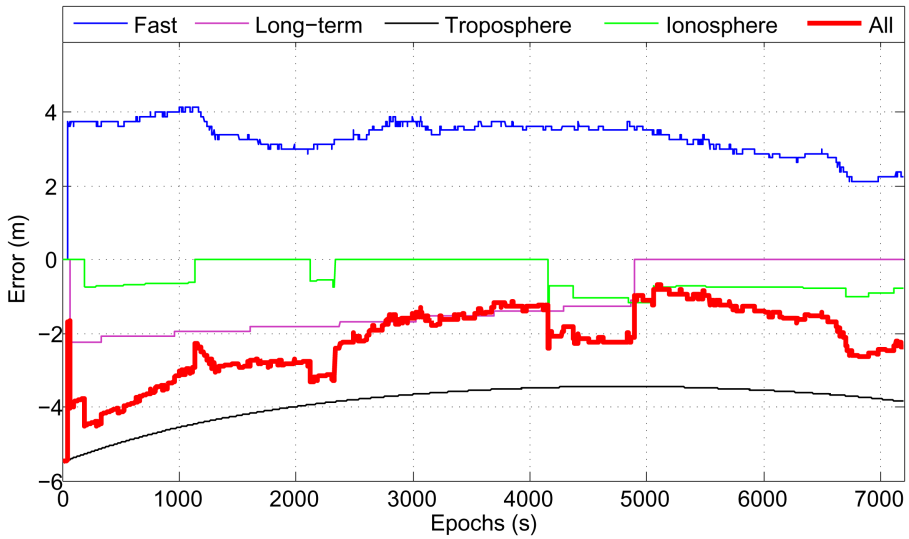


Figure 6.5. Pseudorange corrections

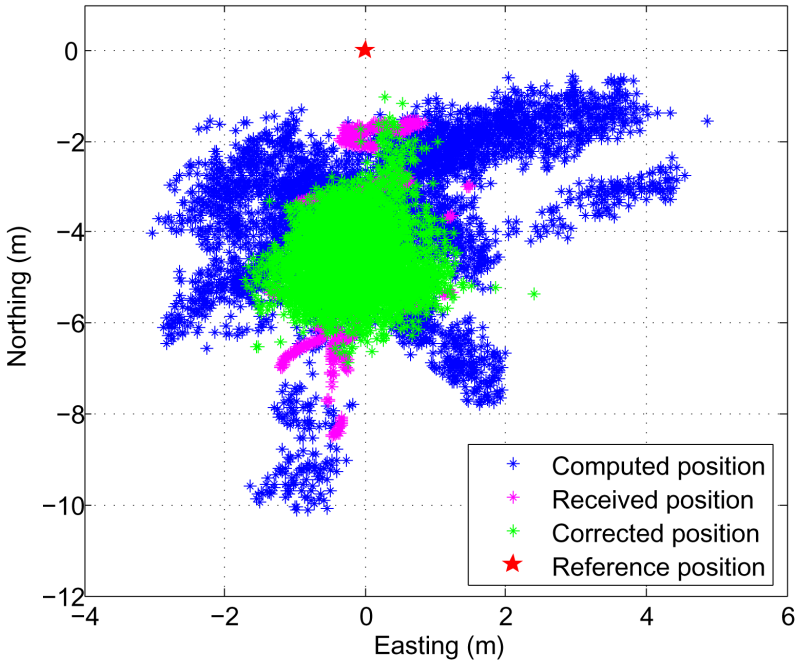


Figure 6.6. Position after using all error corrections

	Corrected position		Computed position		Received position[2]	
	X	Y	X	Y	X	Y
Min	-1.706	-6.862	-3.028	-10.13	-1.354	-8.483
Max	2.396	-1.033	4.86	-0.574	1.501	–1.539
Mean	-0.1412	-4.546	0.2442	-3.689	-0.2884	-4.038
Median	-0.1601	-4.596	0.181	-3.52	-0.2816	-3.974
Std[3]	0.4921	0.7527	1.363	1.576	0.4405	1.65
Range	4.102	5.829	7.888	9.557	2.855	6.944

Table 6.2. Result comparison: corrected position with long-term correction (left) and without (right)

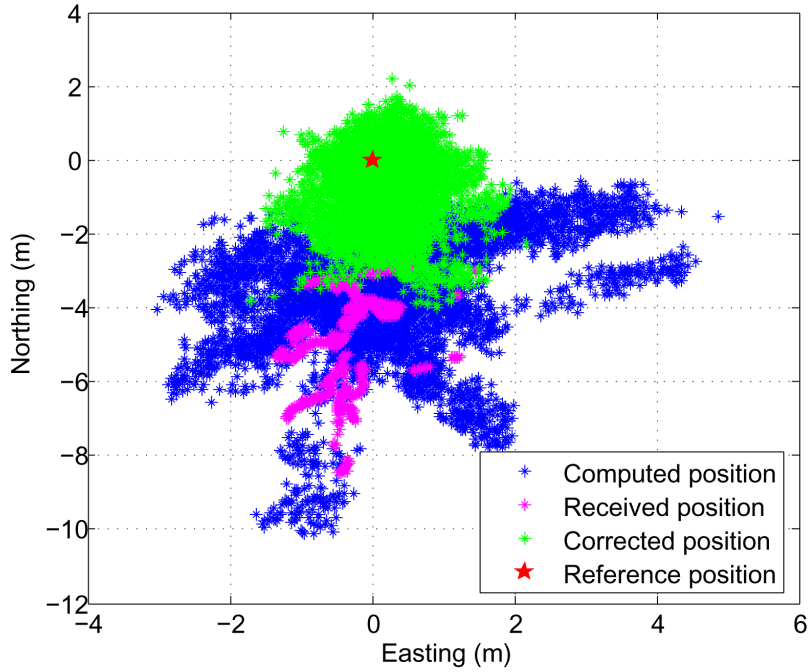


Figure 6.7. Position after removing long-term error corrections

Figure 6.8 with characteristics given in Table 6.4.

The result of using UDREI values degraded so much that it will not be used in computations. The reason for this could be many unclarities in [6] document of how the UDREI values should

[2]Received position is obtained from Thales DG14 GPS receiver
[3]Standard Deviation

	Corrected position			
	With long-term corrections		Without long-term corrections	
	X	Y	X	Y
Min	-1.706	-6.862	-1.728	-3.984
Max	2.396	-1.033	2.158	2.22
Mean	-0.1412	-4.546	0.1962	-0.839
Median	-0.1601	-4.596	0.2075	-0.7384
Std	0.4921	0.7527	0.5197	1.035
Range	4.102	5.829	3.886	6.204

Table 6.3. Result comparison

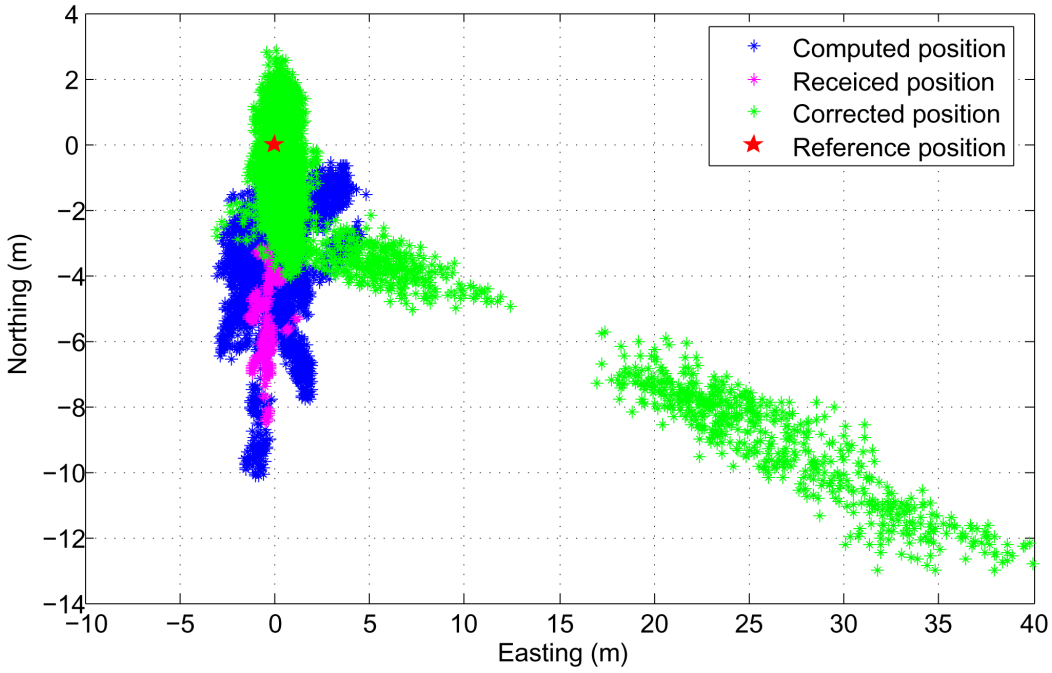


Figure 6.8. Position after using UDREI

be used. The document explains only one case out of many other available how to use UDREI in practice, thus, in order to understand the proper way of correct usage, many tests should be done.

Other problems related with quality control using UDREIs could be mistakes in the code.

Additionally to what it is proposed by [6] document we tried to test position computations

	Corrected position			
	Without UDREI control		With UDREI control	
	X	Y	X	Y
Min	-1.728	-3.984	-3.011	-26.7
Max	2.158	2.22	80.53	2.853
Mean	0.1962	-0.839	4.728	-2.133
Median	0.2075	-0.7384	0.5017	-1.32
Std	0.5197	1.035	12.05	4.123
Range	3.886	6.204	83.54	29.55

Table 6.4. Result comparison

with RRC and without RRC. This idea came to us after reading literature, where some scientists investigated the need of RRC correction computations [23]. To test how RRC is influencing our results we prepared a few figures showing the difference. Figure 6.9 shows the total fast correction without RRC and with RRC applied. From this figure one can see that when RRC is used in the solution it provides some irregularities. The purpose of RRC is mainly to predict the error, especially if it grows or falls constantly in one direction. In such case RRC will improve the output. However, in Figure 6.9 one can see, that the error fluctuate a lot and, thus, applying the RRC may not improve anything and in the contrary can make things even worse. The Figure 6.10 shows the stand-alone RRC. The bounds for RRC amplitude during 2 hour period was from -0.3 to 0.25 meters in pseudorange, which can have some influence in the final position computation.

The final results were obtained by plotting receiver position when RRC is used and when it is not (Figure 6.11). As one can see there are no many differences in solution with RRC (blue dots) and without RRC (green dots) and we can not say that using position computations without RRC can improve the solution. However, the comparison of characteristics showed that according to the statistics there is a very small improvement (millimeter level). The final decision to use RRC or not to use can not be made without more tests, thus, in the final solution RRC correction will be used as it is required by [6].

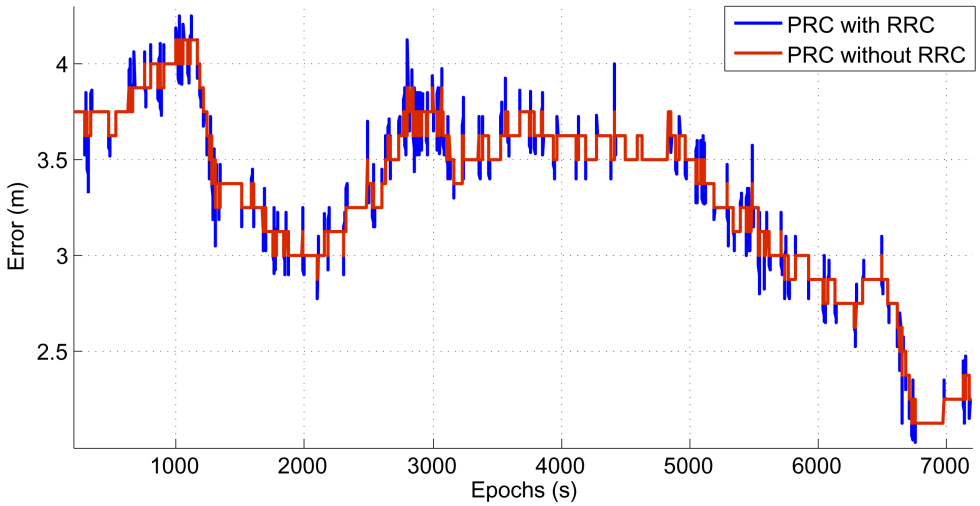


Figure 6.9. PRC with and without RRC correction

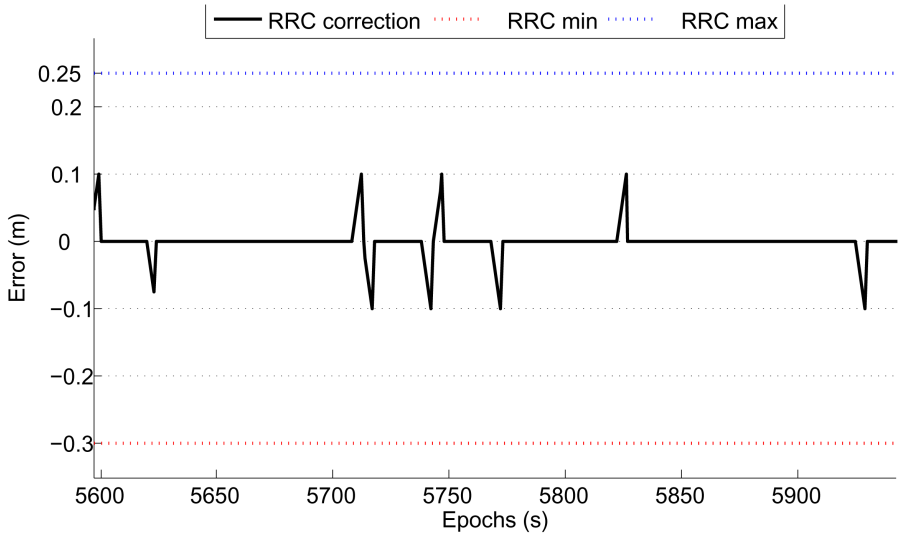


Figure 6.10. RRC error and its bounds

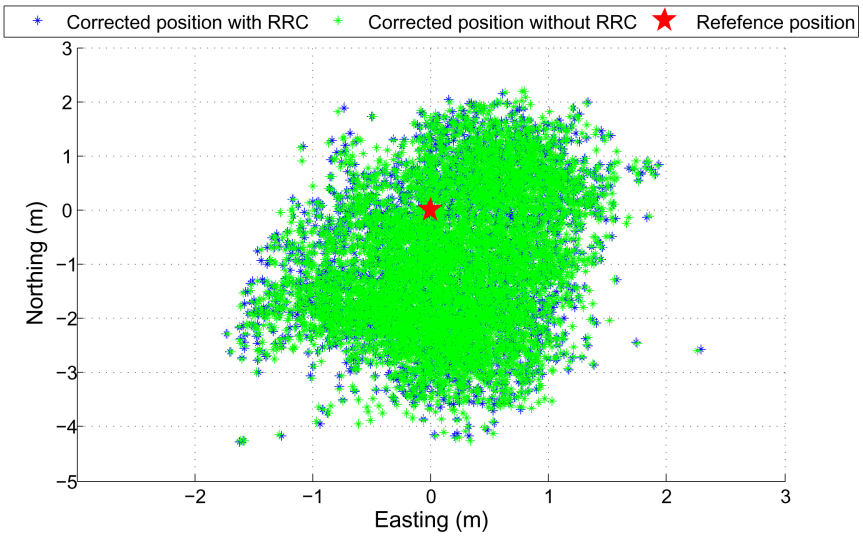


Figure 6.11. RRC influence to position

6.3 Summary

Two different implementations of acquisition algorithm were tested and compared. Modified averaging correlator performed well and showed computation time improvements. All acquired satellites were tracked, except one. The performance of GPS signal tracking was acceptable while the EGNOS signal tracking requires improvements. In data processing part the broad testing procedures were made to validate the output of position computations. Tests presented here cover only the small part of the overall testing procedures. However, it is obvious that it is not enough and further investigations should be organized in order to explain uncertainties obtained by using long-term corrections and UDREIs. Finally, the corrections are within the accuracy limits defined by EGNOS.

Chapter 7

Future Perspectives

The primary objective of this project was to gain insight about how the GPS receiver is actually processing the signal and how the SBAS signal differs from the GPS one and provides the error correction, and to learn the associated signal processing techniques to solve well known existing algorithms. Limited scope was defined keeping the completion time in view and therefore, the receiver was implemented in post-processing. A number of developments, that could be made in this receiver, are listed below:

Real-time Processing: Real-time operation, requiring a high dynamic range and a large bandwidth, the software receiver and the host processor may not have the capabilities to provide true real-time performance. However, the use of pre-processors such as FPGA correlators (for correlation processing) and DSP processors for FFT processing are available and provide real-time capabilities. These hardwares are fully compatible with the RF front ends and can be used as a direct replacement for the software correlators available with the Software GPS Receivers.

Galileo Feature: Signal acquisition and tracking were done for GPS and EGNOS satellites, however the implementation regarding Galileo signals could be left as future development.

Tracking performance improvements : The tracking loops accuracy could be improved. The parameters of the PLL, DLL should be adjusted to decrease the tracking loop errors. The parameters should be changed adaptively with respect to the receiver status, enhancing its capability to handle either low or high dynamics.

Application of new algorithms : Recently many new CDMA signal processing algorithms were developed. More promising algorithms could be implemented. This would improve the overall performance of the software receiver. The receiver then could be upgraded from a prototype version to a fully functional receiver.

Position Accuracy and Precision Improvements: The obtained accuracy and precision of GPS receiver position can be improved by using pseudorange smoothing with carrier phase,

various filtering techniques or weighting position computation with EGNOS quality data. Moreover, the currently used computations can be debugged.

Chapter 8

Conclusions and Recommendations

The software GPS receiver can be configured, modified, tested and evaluated easily. A prototype EGNOS receiver that runs in post process mode was developed and verified in this project. The amount of the computational burden was also optimized with different methods.

Acquisition and tracking results were also presented. Acquisition algorithms performed well. However, no statistical considerations were done in order to determine false alarm probability, and probability of signal detection. Only visual approvement was done, by investigating the acquisition plots. The algorithm performed well, and gave correct detection.

It was evident that GPS and EGNOS signals acquisition performance is acceptable. EGNOS performance tracking was not satisfactory enough. It is a multichannel receiver that operates in post-process mode. Like a conventional receiver it includes several functional blocks: acquisition, code and carrier tracking. Other blocks like bit synchronization, frame/subframe synchronization are left for future refinements.

The position was verified by comparing the known reference point position to the one computed without applying EGNOS corrections and to the one computed with EGNOS corrections. The tests showed the outstanding performance. The results of position computation were improved in accuracy as well as in precision aspects.

If EGNOS tracking performance is improved and enough recorded data is available, then it will be possible to combine the two modules that were implemented. These are: CDMA signal processor, which was able to acquire and track satellites and EGNOS corrections application simulator, which provided error corrected GPS solution. The software receiver can then be verified by testing against real EGNOS/GPS receiver.

Appendix A

Matlab Source Code

A.1 FFT-Based Frequency Domain Acquisition

```
% FFT-Based Acquisition
function [acquired,ph,freq] = acq(fid)
tic %
load carrier.mat %
load codes.mat %
data = fread(fid,119990,'schar');

fc = 3.563e6-10e3:500:3.563e6+10e3; %
acquired = zeros(1,36); %
ph = zeros(1,36); %
freq = zeros(1,36); %
Y = zeros(10,11999*41); %
for fr=1:41
  for hh=1:10
    sv_up = double(data(hh*11999-11998:11999*hh))';
    I = sine(fr,:).*sv_up;
    Q = cosine(fr,:).*sv_up;
    X = fft(I+j*Q);
    Y(hh,fr*11999-11998:fr*11999) = X;
  end
end

TH = [16 11 9 8 8 7 7 7 6 6]; %
for n=1:35
  R    = zeros(41,11999);
```

```
code = codes(n,:);
lcf  = conj(fft(code));
for hh=1:10
  for fr=1:41
    R(fr,:) = R(fr,:)+abs(ifft(Y(hh,fr*11999-11998:fr*11999).*lcf)).^2;
  end
  C  = R;
  TS = max(max(R))/mean(mean(R));
  if TS>TH(1,hh)
    [peak phase] = max(max(R));
    [G_P fr]     = max(max(R'));
    R(fr,:)      = 0;
    %Remove adjacent maximas
    [G_PP frr]   = max(max(R'));
    R(frr,:)     = 0;

    [G_PPP frrr] = max(max(R'));
    R(frrr,:)    = 0;
    [G_PPP frrr] = max(max(R'));
    R(frrr,:)    = 0;
    [G_PPP frrr] = max(max(R'));
    R(frrr,:)    = 0;
    [G_R fr1]    = max(max(R'));
    R(fr1,:)     = 0;
    [G_L fr2]    = max(max(R'));
    F = 5*((G_L-G_R)/G_P);
    m_F   = -4.135*(F^3)+204.5*F;
    fp    = fc(fr);
    f_acq = fp+round(m_F);

    % Fine frequency estimation
    ph(:,n)   = phase;
    freq(:,n) = f_acq;
    acquired(:,n) = n;
    break;
  else
    ph(:,n)   = 0;
    freq(:,n) = 0;
  end;
end
```

```
end

acquired(acquired==0)   = [];
%acquired(acquired==34) = acquired(acquired==34)+3;
%acquired(acquired==35) = acquired(acquired==35)+4;
ph(ph==0)     = []; %
freq(freq==0) = []; %
toc
```

A.2 Modified Averaging Correlator Based Acquisition

```
% Modified Averaging Correlator Based Acquisition
function [acquired,ph,freq]=mod_corr() %
tic %
load data_long.mat %
load carrier.mat %
load codes.mat

downrate = 11999/4096; %
i        = 1:4096; %
fc       = 3.563e6-10e3:500:3.563e6+10e3; %
acquired = zeros(1,36); %
ph       = zeros(1,36); %
freq     = zeros(1,36); %
Y  = zeros(10,4096*41); %
for fr=1:41
   for hh=1:10
      sv_up=double(data(hh*11999-11998:11999*hh))';
      I = sine(fr,:).*sv_up;
      Q = cosine(fr,:).*sv_up;
      I_av(i) = I(ceil(i*downrate));
      Q_av(i) = Q(ceil(i*downrate));
      X = fft(I_av+j*Q_av);
      Y(hh,fr*4096-4095:fr*4096) = X;
  end
end

TH = [16 11 9 8 8 7 7 7 6 6]; %
for n=1:35
  R      = zeros(41,4096);
```

```
  code   = codes(n,:);
  ca(i)  = code(ceil(i*downrate));
  lcf    = conj(fft(ca));
  for hh = 1:10
    for fr=1:41
      R(fr,:) = R(fr,:)+abs(ifft(Y(hh,fr*4096-4095:fr*4096).*lcf)).^2;
    end
    TS = max(max(R))/mean(mean(R));
    if TS>TH(1,hh)
      [peak phase] = max(max(R));
      [peak frequency] = max(max(R'));
      ph(:,n) = round(phase*(11999/4096));
      freq(:,n) = fc(frequency);
      acquired(:,n) = n;
      break;
    else
      ph(:,n)=0;
      freq(:,n)=0;
    end;
  end
end

acquired(acquired==0)  = []; %
acquired(acquired==34) = acquired(acquired==34)+3; %
acquired(acquired==35) = acquired(acquired==35)+4; %
ph(ph==0)              = []; %
freq(freq==0)          = []; %
toc
```

A.3 Tracking Module

```
% Tracking Module
function Ip = track(fid,fc,ph,num_sec,sat_id) %
load codes.mat %
chip_delay = 5; %
scode      = 11999; %
num_to_av  = 20;

fseek(fid,ph,-1); y(1)  = 0; %
pnco  = 0.0; %
```

```
t_end = 0;

% Filter coefficients
B1   = 250; %
damp = 0.707; %
wn   = 2*B1/(damp+1/(4*damp)); %
dT   = 1e-3; K    = 400*pi;

c1 = 1/K*8*damp*wn*dT/(4+4*damp*wn*dT+(wn*dT).^2); %
c2 = 1/K*4*(wn*dT).^2/(4+4*damp*wn*dT+(wn*dT).^2);

%generate CA codes
promt = codes(sat_id,:); %
early = [promt(scode+1-chip_delay:scode)
promt(1:scode-chip_delay)]; %
late = [promt(chip_delay+1:scode) promt(1:chip_delay)];

% Loop
for i=1:round(num_sec*1000/num_to_av)
  for j=1:num_to_av
    t = t_end:(1/11.999e6):(scode-1)/11.999e6+t_end;
    t_end = t(scode)+1/11.999e6;
    Icomp_lo = sin(fc*2*pi*t+pnco);
    Qcomp_lo = cos(fc*2*pi*t+pnco);
    data = fread(fid,scode,'schar')';
    Icomp = data.*Icomp_lo;
    Qcomp = data.*Qcomp_lo;
    index = (i-1)*num_to_av+j;
    Ie(index) = sum(early.*Icomp);
    Ip(index) = sum(promt.*Icomp);
    Il(inda)  = sum(late.*Icomp);
    Qe(index) = sum(early.*Qcomp);
    Qp(index) = sum(promt.*Qcomp);
    Ql(index) = sum(late.*Qcomp);

    ee(inda) = (Ie(inda).^2+Qe(inda).^2).^.5;
    pp(inda) = (Ip(inda).^2+Qp(inda).^2).^.5;
    ll(inda) = (Il(inda).^2+Ql(inda).^2).^.5;

    % Costas loop for carrier tracking
```

```
    % Atan discriminator
    disc(index) = atan(Qp(inda)/Ip(inda));

    % Loop filter
    if index<2
      y(index) = 0;
    else
      y(index) = y(index-1)+(c1+c2)*disc(index)-c1*d2(index-1);

      % Phase for Carrier DCO
      pnco = rem(K*y(index)+pnco,2*pi);
    end
  end

  % Code loop discriminator
  E_L = sum(ee((inda-num_ava+1):inda)./ll((inda-num_ava+1):inda))/num_ava;

  if E_L<.8
    promt = [promt(5:scode) promt(1:4)];
    early = [early(5:scode) early(1:4)];
    late  = [late(5:scode) late(1:4)];
  elseif E_L>1.5
    promt = [promt(scode) promt(1:scode-1)];
    early = [early(scode) early(1:scode-1)];
    late  = [late(scode) late(1:scode-1)];
  end
end
```

A.4 Message Type 18 Parsing Function

```
function a = msg18(mes)
%convert from binary string to binary array
mesbin      = mes';                          %
mesbin      = hex2dec(mesbin);               %
mesbin      = mesbin';

a.Number_of_bands = two_comp(mes(1:4) ,0);  %
a.Band_number     = two_comp(mes(5:8) ,0);  %
a.IODI            = two_comp(mes(9:10),0);  %
a.IGP_Mask        = mesbin(11:211);         %
```

```
a.Spare           = mes(212);
```

A.5 Coordinate Transformation From ECEF (X,Y,Z) To Geodetic (Longitude, Latitude, Altitude)

```
function a = XYZ2LLA(X, Y, Z)
MajorA = 6378137.0;             % ellipsoid major axis
MinorB = 6356752.3142;          % ellipsoid minor axis

% Compute radius of parallel and Theta  and eccentricities
RadiusOfParallel = norm([X,Y],2); %
Theta            = atan2(Z*MajorA, RadiusOfParallel * MinorB); %
Esqr             = (MajorA^2 - MinorB^2) / MajorA^2; %
E2sqr            = (MajorA^2 - MinorB^2) / MinorB^2;

% Compute Latitude and Longitude
a.Latitude  = atan2(Z + (E2sqr * MinorB * sin(Theta)^3),
RadiusOfParallel - (Esqr * MajorA * cos(Theta)^3)); %
a.Longitude = atan2(Y,X);

% Compute the prime vertical curvature
NRadius  =  MajorA^2 / sqrt(MajorA^2 * cos(a.Latitude)^2 +
MinorB^2 * sin(a.Latitude)^2);

% Compute height
a.Altitude = (RadiusOfParallel / cos(a.Latitude)) - NRadius;

% Convert to degrees
a.Longitude = a.Longitude * (180 / pi); %
a.Latitude  = a.Latitude  * (180 / pi);
```

A.6 Ionosphere Error Computation Algorithm

```
function a = iono_new(satx, saty, satz, latitude, longitude, mask,
corr, corr1, el, az);

% Computing Pierce Point
z = PiercePoint(latitude, longitude, el, az, 1);

% closest IGP point west to the pierce point
```

```
k  = z.lat_pp  - mod(z.lat_pp, 5);
% closest IGP point north to the pierce point
k1 = z.long_pp - mod(z.long_pp,5);

% latitude and longitude relative coordinates in mask, corr and corr1 array
% of one of the closest points to the pierce point
rel_lat  = k  / 5 + 18; rel_long = k1 / 5 + 13;

% longitudes and latitudes of selected grid points (4x4)
lat_matrix  = (ones(4,4) * k)  + [-5, -5, -5, -5; 0, 0, 0, 0; 5,
5, 5, 5; 10, 10, 10, 10]; %
long_matrix = (ones(4,4) * k1) + [-5, 0, 5, 10; -5, 0, 5, 10; -5,
0, 5, 10; -5, 0, 5, 10];

% mask for selected grid points
maskx  =  mask((rel_lat-1):(rel_lat+2),(rel_long-1):(rel_long+2));

% IGP Vertical Delay Estimates for selected points
corrx  =  corr((rel_lat-1):(rel_lat+2),(rel_long-1):(rel_long+2));

% sigma^2 GIVE for selected points
corr1x = corr1((rel_lat-1):(rel_lat+2),(rel_long-1):(rel_long+2));

% selected points with IGP VDE and sigma^2 GIVE
matrix   = corrx  .* maskx; %
matrix1  = corr1x .* maskx; %

% Interpolation
int = Interpolation(z.lat_pp, z.long_pp, lat_matrix, long_matrix,
matrix, matrix1);

R_e = 6378136.3;  % Approximate radius of the earth's ellipsoid in meters
h_I = 350000;     % The hight of the maximum electron density in meters

% Obliquity factor
F_pp = (1 - ((R_e * cos(el)) / (R_e+h_I))^2)^(-0.5);

% Ionospheric delay correction
a.IC   = - F_pp   * int.corr;
```

```
% sigma^2 UIRE
a.UIRE =   F_pp^2 * int.UIVE;
```

A.7 Pierce Point Computation Algorithm

```
function a = PiercePoint(Lat_u, Long_u, E, A, out)

R_e = 6378136.3;  % Approximate radius of the earth's ellipsoid in meters
h_I = 350000;     % The hight of the maximum electron density in meters

% The earth's central angle between the user position and the earth
% projection of the pierce point
psi_pp = pi/2 - E - asin((R_e/(R_e + h_I)) * cos(E));

% Pierce Point latitude
phi_pp = asin(sin(Lat_u* (pi/180))*cos(psi_pp) + cos(Lat_u*
(pi/180))*sin(psi_pp)*cos(A));

% Pierce Point longitude
if ((Lat_u > 70) && (tan(psi_pp)*cos(A) > tan(pi/2 - Lat_u*
(pi/180)))) || ((Lat_u < -70) && (tan(psi_pp)*cos(A+pi) > tan(pi/2
+ Lat_u* (pi/180))))
    lamda_pp = Long_u* (pi/180) + pi - asin((sin(psi_pp)*sin(A)) / cos(phi_pp));
else
    lamda_pp = Long_u* (pi/180) + asin((sin(psi_pp)*sin(A)) / cos(phi_pp));
end

% set output format
if out == 1
    a.lat_pp  = phi_pp   * (180/pi);
    a.long_pp = lamda_pp * (180/pi);
else
    a.lat_pp  = phi_pp;
    a.long_pp = lamda_pp;
end
```

A.8 Function To Check If Point Is In Triangle

```
% Function checks if the point is inside the given triangle
%
```

```
% Input:
%    x1,x2         - x and y coordinates of triangle's first angle
%    y1,y2         - x and y coordinates of triangle's second angle
%    z1,z2         - x and y coordinates of triangle's third angle
%    point1,point2 - x and y coordinates of point we want to check
% Output:
%    status        - the status of computation
%                      0  -  if point is outside triangle
%                      1  -  if point is inside triangle
function status = triangle (x1, x2, y1, y2, z1, z2, point1,point2)%
g  = (point1 - x1) * (y2 - x2) - (point2 - x2) * (y1 - x1); %
g1 = (point1 - y1) * (z2 - y2) - (point2 - y2) * (z1 - y1); %
g2 = (point1 - z1) * (x2 - z2) - (point2 - z2) * (x1 - z1); %

if g >= 0 && g1 >= 0 && g2 >= 0
    status = 1;
else
    status = 0;
end
```

A.9 Function To Compute Troposphere Delay Error

```
% Input:
%    day       - Day of the year
%    latitude  - user latitude
%    elevation - satellite elevation angle
%    H         - hight above mean see level
%
% Output:
%    a.corr    - The tropospheric delay estimate
%    a.error   - Residual Tropospheric Error

function a = tropo_new(day, latitude, elevation, H)

% The tropospheric correction mapping function for satellite elevation

if latitude > 0
    D_min = 28;
else
    D_min = 211;
```

```
end

latitudex = abs(latitude);
% Meteorological parameters for tropospheric delay
%                P_o      T_o     e_o     B_o     l_o
    average  = [1013.25, 299.65, 26.31, 0.0063,  2.77;
                1017.25, 294.15, 21.79, 0.00605, 3.15;
                1015.75, 283.15, 11.66, 0.00558, 2.75;
                1011.75, 272.15,  6.78, 0.00539, 1.81;
                1013.0,  263.65,  4.11, 0.00453, 1.55];
%                  dP      dT      de     dB      dl
    seasonal = [   0.0,    0.0,   0.0,  0.0,     0.0;
                  -3.75,   7.0,   8.85, 0.00025, 0.33;
                  -2.25,  11.0,   7.24, 0.00032, 0.46;
                  -1.75,  15.0,   5.36, 0.00081, 0.74;
                  -0.5,   14.5,   3.39, 0.00062, 0.3];

if latitudex <= 15
    averagex  = average(1,:);
    seasonalx = seasonal(1,:);
elseif latitudex > 15 & latitudex < 30
        averagex  = average(1,:)  + (average(2,:)  - average(1,:))  *
        ((latitudex - 15)/15);
        seasonalx = seasonal(1,:) + (seasonal(2,:) - seasonal(1,:)) *
        ((latitudex - 15)/15);
elseif latitudex == 30
    averagex  = average(2,:);
    seasonalx = seasonal(2,:);
elseif latitudex > 30 & latitudex < 45
        averagex  = average(2,:)  + (average(3,:)  - average(2,:))  *
        ((latitudex - 30)/15);
        seasonalx = seasonal(2,:) + (seasonal(3,:) - seasonal(2,:)) *
        ((latitudex - 30)/15);
elseif latitudex == 45
    averagex  = average(3,:);
    seasonalx = seasonal(3,:);
elseif latitudex > 45 & latitudex < 60
        averagex  = average(3,:)  + (average(4,:)  - average(3,:))  *
        ((latitudex - 45)/15);
        seasonalx = seasonal(3,:) + (seasonal(4,:) - seasonal(3,:)) *
```

```
        ((latitudex - 45)/15);
elseif latitudex == 60
    averagex  = average(4,:);
    seasonalx = seasonal(4,:);
elseif latitudex > 60 & latitudex < 75
        averagex  = average(4,:)  + (average(5,:)  - average(4,:))  *
        ((latitudex - 60)/15);
        seasonalx = seasonal(4,:) + (seasonal(5,:) - seasonal(4,:)) *
        ((latitudex - 60)/15);
elseif latitudex >= 75
    averagex  = average(5,:);
    seasonalx = seasonal(5,:);
end

final = averagex - seasonalx * cos((2*pi*(day-D_min))/365.25);

k_1 = 77.604;  % K/mbar
k_2 = 382000;  % K^2/mbar
R_d = 287.054; % J/kg/K
g_m = 9.784;   % m/s^2
% Zero-altitude zenith delay terms
Z_h = (10^(-6) * k_1 * R_d * final(1)) / g_m; %
Z_w = ((10^(-6)* k_2 * R_d) / (g_m * (final(5)+1) - final(4) *
R_d)) * (final(3)/final(2));

g   = 9.80665; % m/s^2
% d_h and d_w
d_h = (1 - (final(4) * H) / final(2))^(g / (R_d * final(4))) *
Z_h; %
d_w = (1 - (final(4) * H) / final(2))^((((final(5) + 1)* g) /
(R_d * final(4))) - 1) * Z_w;

% a_hyd a_wet, b_hyd b_wet and c_hyd c_wet
abc_hyd = [(1.18972 - 0.026855 * (H/1000) + 0.10664 *
cos(latitude))* 0.001 , 0.0035716 , 0.082456 ]; %
abc_wet = [(0.61120 - 0.035348 * (H/1000) - 0.01526 *
cos(latitude))* 0.001 , 0.0018576 , 0.062741 ];

temp = sin(elevation);
% m_h and m_w
```

```
m_h = (1 + (abc_hyd(1) / (1 + (abc_hyd(2) / (1 + abc_hyd(3)))))) /
(temp + (abc_hyd(1) / (temp + (abc_hyd(2) / (temp +
abc_hyd(3)))))); %
m_w = (1 + (abc_wet(1) / (1 + (abc_wet(2) / (1 +
abc_wet(3)))))) / (temp + (abc_wet(1) / (temp + (abc_wet(2) /
(temp + abc_wet(3)))))));

% The tropospheric delay estimate
a.corr  = -(d_h * m_h + d_w * m_w);

% Tropospheric vertical error
sigma_TVE = 0.12;

% Residual Tropospheric Error
a.error = sigma_TVE * ((m_h + m_w) / 2);
```

Appendix B

Working Process

This chapter describes the organization and management of this whole project. It contains group management including task assignments, time management and binding the group together.

B.1 Task division

After realizing the goal of this project precisely, a group leader was chosen and weekly meetings of the group were conducted for discussions. During initial discussions it was decided to split the group into two subgroups: one was responsible for the decoding and applying error correction and the other was responsible for acquisition, tracking and demodulation of EGNOS signals.

B.2 Group Organization and Working Environment

As mentioned in the previous section the group was divided into two subgroups, a group leader for either of the group was chosen to ensure better management. The working environment has been good throughout the project duration, however, sometimes the regular daily meeting in the group room could not be conducted. The coordination between the group members was good.

B.3 Time Management

Task assignments and time management were managed by the group leader. A time plan was made in which the tasks and subtasks were defined and the time resource was allocated to each task. The major tasks were studying, report writing, analysis, design and implementation and testing. The Gantt chart shows task and subtask and their associated time schedule. The study and report writing phase were started early. The implementation part was subdivided into acquisition, tracking and application of EGNOS error correction and testing.

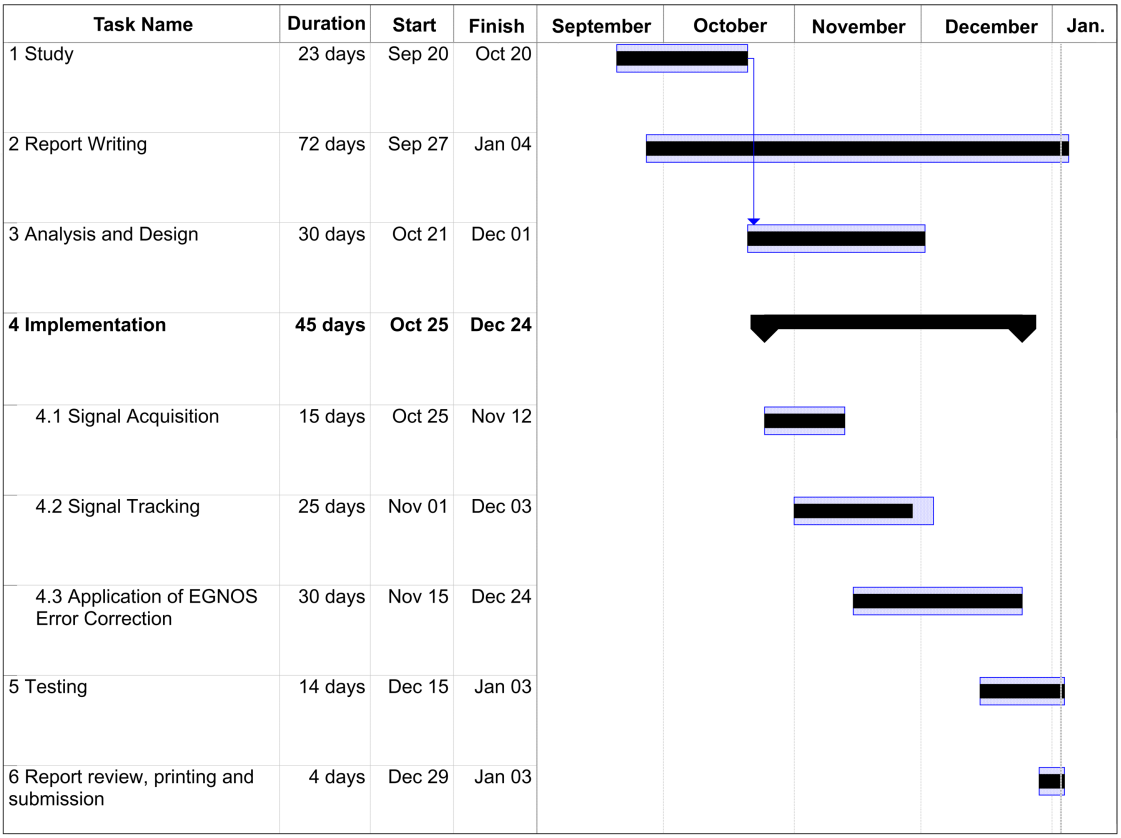


Figure B.1. Project schedule

The time management went well, however there was deviation from the initial time plans and therefore most of the work focus was on documentation during the last month.

B.4 Resource Management

The resources required in this project were recorded data from any EGNOS enabled GPS receiver or EMS server, time and human resource. The data was recorded using an existing EGNOS receiver.

B.5 Problems

One of the major problem was management of a big group, It was sometimes difficult to have regular daily meetings with every group member. However, tasks were assigned weekly and the group leader was reported about it.

B.6 Suggestions

Some useful suggestions are listed below on the basis of our work experience in the group this semester:

Group Size: It is suggested to keep the group size smaller to have effective coordination and results.

Regular Meetings: Meetings should be conducted week or biweekly depending on the progress and requirement of work.

Organization The organization should be implemented in a way that there should be centralized control. However, to avoid conflicts the group leaders should report to other members to keep the working environment friendly and effective.

Appendix C

EGNOS System Operational Status

The EGNOS system was designed and developed under the responsibility of the European Space Agency (ESA). [2] The very nature of EGNOS - a safety critical satellite navigation system composed of geographically distributed elements - dictated the creation of a centralized coordination company. 6 European Air Traffic Service Providers (NATS, DGAC, AENA, ENAV, DFS and skyguide) founded the European Satellite Services Provider (ESSP) to act as the EGNOS System Operator and Service Provider, i.e. to coordinate EGNOS operations and provide EGNOS data services.

EGNOS system is still not fully operational. The ESSP plan of the EGNOS signal provision is shown in Figure C.1.

Europe faces a major challenge in the creation of a European GNSS institutional framework to support Satellite Navigation Service Provision. The ESSP, with the help of its member organizations and partners, will assist in the development of a European GNSS strategy, to ensure that EGNOS Safety-of-Life services are available as early as possible. The successful introduction of EGNOS is of key importance to the Galileo program. The experience gained with EGNOS should help in the early establishment of Galileo operations and services.

The EGNOS system satellites broadcasting plan is illustrated below[1] (Figure C.2).

[1] Information is on the date of 30 September 2004

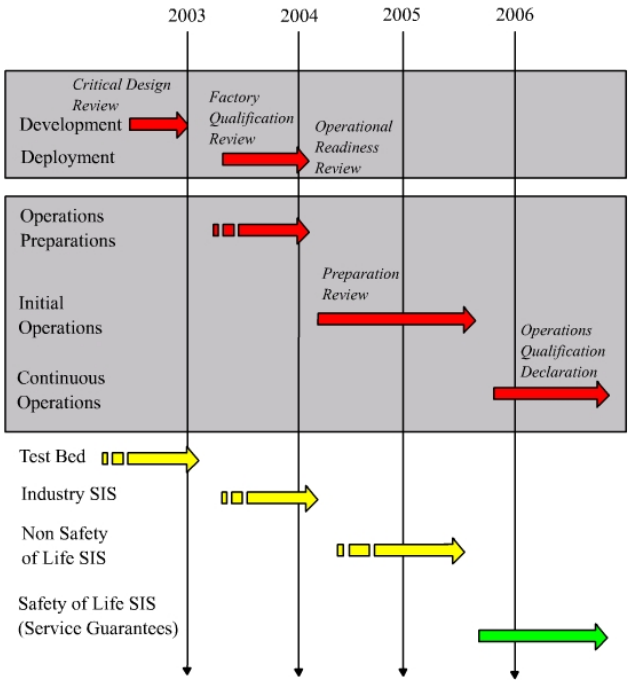


Figure C.1. Planning for ESSP operations and EGNOS SIS Provision

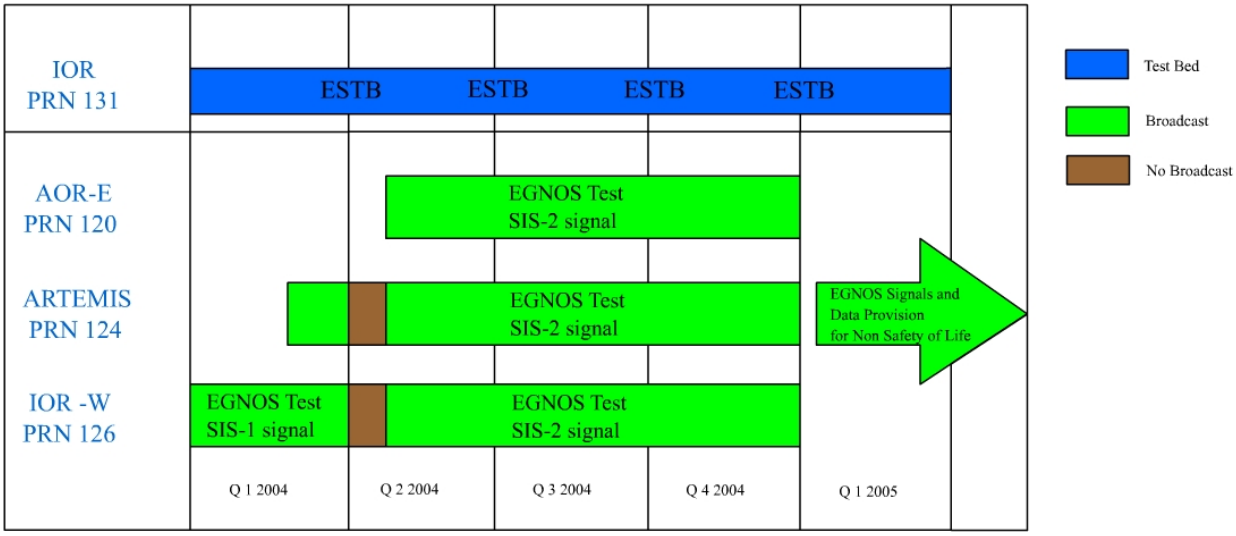


Figure C.2. ESTB and EGNOS satellites expected broadcasting plan

Appendix D

WAAS/EGNOS Data Types

D.1 Tables

Parameter	No.of Bits	Scale Factor (LSB)	Effective Range	Units
IODF	2	1	0 to 3	unitless
IODP	2	1	0 to 3	unitless
PRC (13 x)	12	0.125	-256.0 to +255.875	meters
UDREI (13 x)	4	1	0 to 15	unitless

Table D.1. Fast correction message types 2-5

Type	Contents
0	Don't use for safety applications (for WAAS testing)
1	PRN mask assignments, set up to 51 of 210 bits
2 to 5	Fast corrections
6	Integrity information
7	Fast correction degradation factor
8	Reserved for future messages
9	GEO navigation message (X, Y, Z, time, etc.)
10	Degradation parameters
11	Reserved for future messages
12	WAAS Network Time/UTC offset parameters
13 to 16	Reserved for future messages
17	GEO satellite almanacs
18	Ionospheric grid point masks
19 to 23	Reserved for future messages
24	Mixed fast corrections/long term satellite error corrections
25	Long term satellite error corrections
26	Ionospheric delay corrections
27	WAAS Service Message
28	Clock-Epehemeris Covariance Matrix Message
29 to 61	Reserved for future messages
62	Internal Test Message
63	Null Message

Table D.2. Message types

Parameter	No.of Bits	Scale Factor (LSB)	Effective Range	Units
$IODF_2$	2	1	0 to 3	unitless
$IODF_3$	2	1	0 to 3	unitless
$IODF_4$	2	1	0 to 3	unitless
$IODF_5$	2	1	0 to 3	unitless
For each of 51 satellites	–	–	–	–
UDREI	4	(see Table D.4)	(see Table D.4)	unitless

Table D.3. Type 6 - integrity message content

$\mathbf{UDREI_i}$	$\mathbf{UDREI_i}$ Meters	$\boldsymbol{\sigma^2_{i,UDRE}}$ $\mathbf{Meters^2}$
0	0.75	0.0520
1	1.0	0.0924
2	1.25	0.1444
3	1.75	0.2830
4	2.25	0.4678
5	3.0	0.8315
6	3.75	1.2992
7	4.5	1.8709
8	5.25	2.5465
9	6.0	3.3260
10	7.5	5.1968
11	15.0	20.7870
12	50.0	230.9661
13	150.0	2078.695
14	Not monitored	Not monitored
15	Do not use	Do not use

Table D.4. Evaluation of $UDREI_i$

Parameter	**No. of Bits**	**Scale Factor (LSB)**	**Effective Range**	**Units**
System latency (t_{lat})	4	1	0 to 15	seconds
IODP	2	1	0 to 3	unitless
Spare	2	–	–	–
For each of 51 satellites	204	–	–	–
Degradation Factor indicator (ai_i)	4	1	0 - 15	–

Table D.5. Fast correction degradation factor message contents (message type 7)

Fast Corrections Degradation Factor Indicator (ai_i)	Fast Corrections Degradation Factor (ai) (m/s^2)	User Time-Out Interval for fast corrections seconds En Route through Nonprecision Approach (I_{fc})	User Time-Out Interval for fast corrections seconds Precision Approach Mode (I_{fc})	Maximun Fast Correction Update Interval (seconds)
0	0.00000	180	120	60
1	0.00005	180	120	60
2	0.00009	153	102	51
3	0.00012	135	90	45
4	0.00015	135	90	45
5	0.00020	117	78	39
6	0.00030	99	66	33
7	0.00045	81	54	27
8	0.00060	63	42	21
9	0.00090	45	30	15
10	0.00150	45	30	15
11	0.00210	27	18	9
12	0.00270	27	18	9
13	0.00330	27	18	9
14	0.00460	18	12	6
15	0.00580	18	12	6

Table D.6. Fast corrections degradation factor and user time-out interval evaluation

Parameter	No. of Bits	Scale Factor (LSB)	Effective Range	Units
Velocity Code = 0	1	1	–	unitless
PRN Mask No.	6	1	0 to 51	–
Issue of Data	8	1	0 to 255	unitless
δx (ECEF)	9	0.125	±32	meters
δy(ECEF)	9	0.125	±32	meters
δz (ECEF)	9	0.125	±32	meters
δa_{f0}	10	2^{-31}	$\pm 2^{-22}$	seconds
PRN Mask No.	6	1	0 to 51	–
Issue of Data	8	1	0 to 255	unitless
δx (ECEF)	9	0.125	±32	meters
δy(ECEF)	9	0.125	±32	meters
δz (ECEF)	9	0.125	±32	meters
δa_{f0}	10	2^{-31}	$\pm 2^{-22}$	seconds
IODP	2	1	0 to 3	unitless
Spare	1	–	–	–

Table D.7. Type 25 - long-term satellite error corrections half message parameters with Velocity Code of 0

Parameter	No.of Bits	Scale Factor (LSB)	Effective Range	Units
Velocity Code = 1	1	1	–	unitless
PRN Mask No.	6	1	0 to 51	–
Issue of Data	8	1	0 to 255	unitless
δx (ECEF)	11	0.125	±128	meters
δy (ECEF)	11	0.125	±128	meters
δz (ECEF)	11	0.125	±128	meters
δa_{f0}	11	2^{-31}	$\pm 2^{-21}$	seconds
δx rate-of-change (ECEF)	8	2^{-11}	±0.0625	meters/sec
δy rate-of-change (ECEF)	8	2^{-11}	±0.0625	meters/sec
δzrate-of-change (ECEF)	8	2^{-11}	±0.0625	meters/sec
δa_{f1}	8	2^{-39}	$\pm 2^{-32}$	seconds/sec
Time-of-Day Applicability t_0	13	16	0 to 86,384	seconds
IODP	2	1	0 to 3	unitless

Table D.8. Type 25 - long-term satellite error corrections half message parameters with Velocity Code of 1

Number of Band		Bits in Mask
Band 3		
60 W	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N,65N, 75N	1 to 27
55 W	55S, 50S, 45S, ..., 45N, 50N, 55N	28 to 50
50 W	85S, 75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	51 to 78
45 W	55S, 50S, 45S, ..., 45N, 50N, 55N	79 to 101
40 W	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	102 to 28
35 W	55S, 50S, 45S, ..., 45N, 50N, 55N	129 to 151
30 W	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	152 to 178
25 W	55S, 50S, 45S, ..., 45N, 50N, 55N	179 to 201
Band 4		
20 W	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	1 to 27
15W	55S, 50S, 45S, ..., 45N, 50N, 55N	28 to 50
10 W	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	51 to 77
5W	55S, 50S, 45S, ..., 45N, 50N, 55N	78 to 100
0	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N, 85N	101 to 128
5 E	55S, 50S, 45S, ..., 45N, 50N, 55N	129 to 151
10 E	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	152 to 178
15 E	55S, 50S,45S, ..., 45N, 50N, 55N	179 to 201
Band 5		
20 E	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	1 to 27
25E	55S, 50S, 45S, ..., 45N, 50N, 55N	28 to 50
30 E	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	51 to 77
35E	55S, 50S, 45S, ..., 45N, 50N, 55N	78 to 100
40 E	85S, 75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	101 to 128
45 E	55S, 50S, 45S, ..., 45N, 50N, 55N	129 to 151
50 E	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	152 to 178
55 E	55S, 50S, 45S, ..., 45N, 50N, 55N	179 to 201
Band 6		
60 W	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	1 to 27
65W	55S, 50S, 45S, ..., 45N, 50N, 55N	28 to 50
70 W	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	51 to 77
75W	55S, 50S, 45S, ..., 45N, 50N, 55N	78 to 100
80 E	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N	101 to 127
85 E	55S, 50S, 45S, ..., 45N, 50N, 55N	128 to 150
90 E	75S, 65S, 55S, 50S, 45S, ..., 45N, 50N, 55N, 65N, 75N, 85N	151 to 178
95 E	55S, 50S, 45S, ..., 45N, 50N, 55N	179 to 201

Table D.9. Ionospheric mask bands

Parameter	No.of Bits	Scale Factor (LSB)	Effective Range	Units
Number of Bands being Broadcast	4	1	0 to 11	unitless
Band Number	4	1	0 to 10	unitless
Issue of Data- Ionosphere (IODI)	2	1	0 to 3	unitless
IGP Mask	201	–	–	unitless
Spare	1	–	–	–

Table D.10. Type 18 - IGP mask message contents

Parameter	No.of Bits	Scale Factor (LSB)	Effective Range	Units
Band Number	4	1	0 to 10	unitless
Block ID	4	1	0 to 13	unitless
For each of 15 Grid Points	13	–	–	–
IGP Vertical Delay Estimate	9	0.125	0 to 63.875	meters
Grid Ionospheric Vertical Error Indicator(GIVEI)	4	1	0 to 15	unitless
IODI	2	1	0 to 3	unitless
Spare	7	–	–	–

Table D.11. Ionospheric delay model parameters for message type 26

$\mathbf{GIVEI}_i$	$\mathbf{GIVEI}_i$ Meters	$\delta^2_{i,GIVE}$ $\mathbf{Meters}^2$
0	0.3	0.0084
1	0.6	0.0333
2	0.9	0.0749
3	1.20	0.1331
4	1.5	0.2079
5	1.8	0.2994
6	2.1	0.4075
7	2,4	0.5322
8	2.7	0.6735
9	3.0	0.8315
10	3.6	1.1974
11	4.5	1.8709
12	6.0	3.3260
13	15.0	20.7870
14	45.0	187.0826
15	Not Monitored	Not Monitored

Table D.12. Evaluation of $GIVE_i$

Latitude	Average					Seasonal Variation				
	$\mathbf{P_0}$ (mbar)	$\mathbf{T_0}$ (K)	$\mathbf{e_0}$ (mbar)	β_0 (K/m)	λ_0	ΔP (mbar)	Δt (K)	Δe (mbar)	Db (K/m)	Dl
15° or less	1013.25	229.65	26.31	$6.30e^{-3}$	2.77	0.00	0.00	0.00	$0.00e^{-3}$	0.00
30°	1017.25	294.15	21.79	$6.05e^{-3}$	3.15	-3.75	7.00	8.85	$0.25e^{-3}$	0.33
45°	1015.75	283.15	11.66	$5.58e^{-3}$	2.57	-2.25	11.00	7.24	$0.32e^{-3}$	0.46
60°	1011.75	272.15	6.78	$5.39e^{-3}$	1.81	-1.75	15.00	5.36	$0.81e^{-3}$	0.74
77° or greater	1013.00	263.65	4.11	$4.53e^{-3}$	1.55	-0.50	14.50	3.39	$0.62e^{-3}$	0.30

Table D.13. Meteorological parameters for tropospheric delay

D.2 Ionospheric Grid Point Selection

1. For an IPP between N60^o and S60^o:
   (a) if four IGPs that define a 5-degree-by-5-degree cell around the IPP are set to one in the IGP mask, they are selected; else,
   (b) if any three IGPs that define a 5-degree-by-5-degree triangle that circumscribes the IPP are set to one in the IGP mask, they are selected; else,
   (c) if any four IGPs that define a 10-degree-by-10-degree cell around the IPP are set to one in the IGP mask, they are selected; else,
   (d) if any three IGPs that define a 10-degree-by-10-degree triangle that circumscribes the IPP are set to one in the IGP mask, they are selected; else,
   (e) an ionospheric correction is not available.
2. For an IPP between N60^o and N75^o or between S60^o and S75^o:
   (a) if four IGPs that define a 5-degree latitude -by-10-degree longitude cell around the IPP are set to one in the IGP mask, they are selected; else,
   (b) if any three IGPs that define a 5-degree latitude -by-10-degree longitude triangle that circumscribes the IPP are set to one in the IGP mask, they are selected; else,
   (c) if any four IGPs that define a 10-degree-by-10-degree cell around the IPP are set to one in the IGP mask, they are selected; else,
   (d) if any three IGPs that define a 10-degree-by-10-degree triangle that circumscribes the IPP are set to one in the IGP mask, they are selected; else,
   (e) an ionospheric correction is not available.
3. For an IPP between N75^o and N85^o or between S75^o and S85^o:
   (a) if two nearest IGPs at 75^o and the two nearest IGPs at 85^o (separated by 30^o longitude if band 9 or 10 is used, separated by 90^o otherwise) are set to one in the IGP mask, a 10-degree-by-10-degree cell is created by linearly interpolating between the IGPs at 8^o to obtain virtual IGPs at longitudes equal to the longitudes of the IGPs at 75^o[1]; else,
   (b) an ionospheric correction is not available.
4. For an IPP north of N85^o:
   (a) if the four IGPs at N85^o latitude and longitudes of W180^o, W90^o, 0^o and E90^o are set to one in the IGP mask, they are selected; else,

[1]The $\delta^2_{GIVE^S}$ are linearly interpolated along the 85 degrees line to form virtual $\delta^2_{GIVE^S}$ to go with the virtual IGPs

(b) an ionospheric correction is not available.

5. For an IPP south of S85^o:

   (a) if the four IGPs at S85^o latitude and longitudes of W140^o, W50^o, E40^o and E130^o are set to one in the IGP mask, they are selected; else,

   (b) an ionospheric correction is not available.

D.3 Interpolation Algorithm Definitions

D.3.1 4 point weighting function

$$\begin{aligned} W_1 &= x_{pp} \cdot y_{pp} \\ W_2 &= (1 - x_{pp}) \cdot y_{pp} \\ W_3 &= (1 - x_{pp}) \cdot (1 - y_{pp}) \\ W_4 &= x_{pp} \cdot (1 - y_{pp}) \end{aligned}$$

Where x_{pp} and y_{pp} for IPP's between N85° and S85° are computed as:

$$\begin{aligned} x_{pp} &= \frac{\Delta\lambda_{pp}}{\lambda_2 - \lambda_1} \\ y_{pp} &= \frac{\Delta\phi_{pp}}{\phi_2 - \phi_1} \end{aligned}$$

$\Delta\lambda_{pp}$ and $\Delta\phi_{pp}$ are expressed as (Figure D.1):

$$\begin{aligned} \Delta\lambda_{pp} &= \lambda_{pp} - \lambda_1 \\ \Delta\phi_{pp} &= \phi_{pp} - \phi_1 \end{aligned}$$

D.3.2 3 point weighting function

$$\begin{aligned} W_1 &= y_{pp} \\ W_2 &= 1 - x_{pp} - y_{pp} \\ W_3 &= x_{pp} \end{aligned}$$

The definitions of x_{pp} and y_{pp} are given previously (Figure D.2).

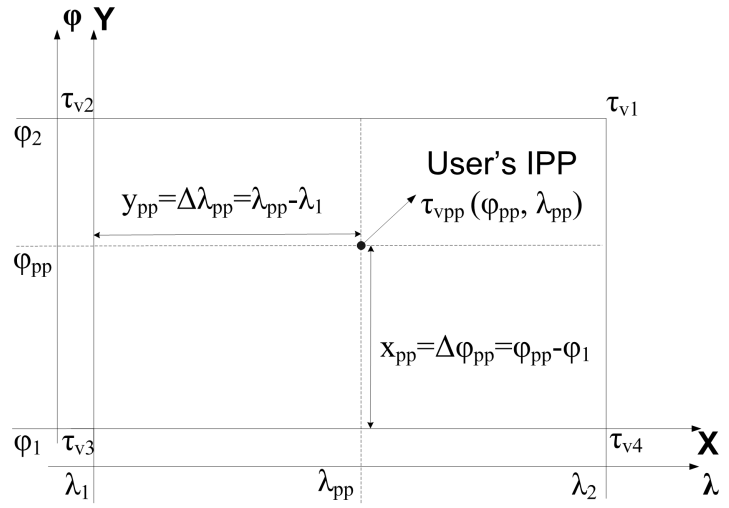


Figure D.1. 4 point interpolation

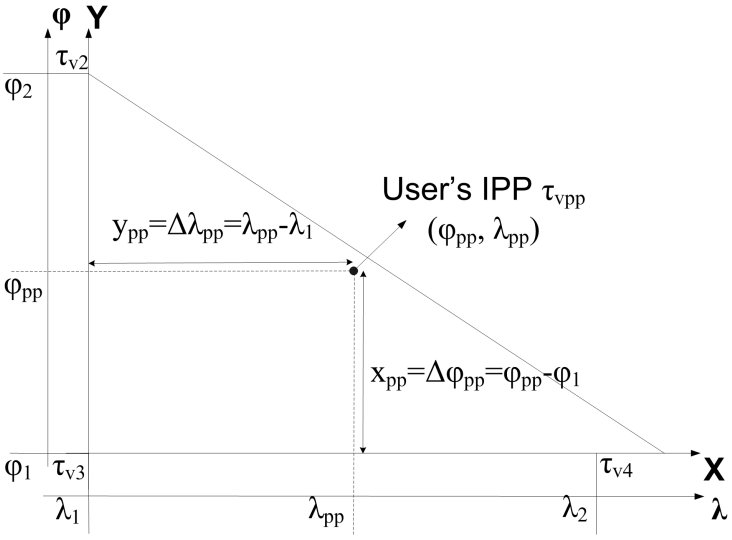


Figure D.2. 3 point interpolation

Appendix E

Algorithms

E.1 Satellite Position Calculation

Satellite position is computed from GPS satellite ephemeris parameters in the navigation message (Table E.1) and few constants (Table E.2).

t_{oe}	Reference time of ephemeris
$\sqrt{A}$	Square root of semi-major axis
e	Eccentricity
i_0	Inclination angle at time t_{oe}
Ω_0	Longitude of the ascending node (at weekly epoch)
ω	Argument of perigee (at time t_{oe})
$\dot{I}$	Rate of change of inclination angle (di/dt)
M_0	Mean anomaly (at time t_{oe})
$\dot{\Omega}$	Rate of change of longitude of the ascending node
Δn	Mean motion correction
C_{uc}	Amplitude of cosine correction to argument of latitude
C_{us}	Amplitude of sine correction to argument of latitude
C_{rc}	Amplitude of cosine correction to orbital radius
C_{rs}	Amplitude of sine correction to orbital radius
C_{ic}	Amplitude of cosine correction to inclination angle
C_{is}	Amplitude of sine correction to inclination angle
T_{GD}	Estimated Group Delay Differential
$IODE$	Issue of Data (Ephemeris)

Table E.1. GPS ephemeris data definitions

The procedure for calculating the satellite position at the time of transmission (T_{tr}) is given in Table E.3.

Constant	Description
$\mu = 3.986005 \cdot 10^{14}$	The earth's universal gravitation constant (m^3/sec^2)
$\dot{\Omega}_e = 7.2921151467 \cdot 10^{-5}$	The earth's rotation rate (rad/sec)
$\pi = 3.1415926535898$	WGS-84 value for π

Table E.2. Constants used in satellite position calculation

(1)	$A = \left(\sqrt{A}\right)^2$	Semi-major axis
(2)	$n = \sqrt{\frac{\mu}{A^3}} + \Delta n$	Corrected mean motion
(3)	$t_k = T_{tr} - t_{oe}$	Time from ephemeris epoch
(4)	$M_k = M_0 + n \cdot t_k$	Mean anomaly
(5)	$M_k = E_k - e \cdot \sin E_k$	Eccentricity anomaly (must be solved Iteratively for E_k)
(6)	$\sin v_k = \frac{\sqrt{1-e^2} \cdot \sin E_k}{1 - e \cdot \cos E_k}$ $\cos v_k = \frac{\cos E_k - e}{1 - e \cdot \cos E_k}$	True anomaly
(7)	$\phi_k = \nu_k + \omega$	Argument of latitude
(8)	$\delta\phi_k = C_{us} \cdot \sin(2\phi_k) + C_{uc} \cdot \cos(2\phi_k)$	Argument of latitude correction
(9)	$\delta r_k = C_{rs} \cdot \sin(2\phi_k) + C_{rc} \cdot \cos(2\phi_k)$	Radius correction
(10)	$\delta i_k = C_{is} \cdot \sin(2\phi_k) + C_{ic} \cdot \cos(2\phi_k)$	Inclination correction
(11)	$u_k = \phi_k + \delta\phi_k$	Corrected argument of latitude
(12)	$r_k = A \cdot (1 - e \cdot \cos E_k) + \delta r_k$	Corrected radius
(13)	$i_k = i_0 + \dot{I} \cdot t_k + \delta i_k$	Corrected inclination
(14)	$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e) \cdot t_k - \dot{\Omega}_e \cdot t_{oe}$	Corrected longitude of node
(15)	$x_p = r_k \cdot \cos u_k$	In-Plane x position
(16)	$y_p = r_k \cdot \sin u_k$	In-Plane y position
(17)	$x_s = x_p \cdot \cos\Omega_k - y_p \cdot \cos i_k \cdot \sin\Omega_k$	ECEF x-coordinate
(18)	$y_s = x_p \cdot \sin\Omega_k + y_p \cdot \cos i_k \cdot \cos\Omega_k$	ECEF y-coordinate
(19)	$z_s = y_p \cdot \sin i_k$	ECEF z-coordinate

Table E.3. Computation of satellite's ECEF position vector

To find E from the 5-th step in the algorithm the iteration is used. The formula can be rewritten as:

$$E_{i+1} = M + e \cdot sin(E_i) \tag{E.1}$$

where

E_{i+1} - the present value
E_i - the previous value

The iteration process should end after the $E_{i+1} - E_i$ is less than a predetermined value (e.g. $1.0 \cdot 10^{-14}$).

E.2 Receiver Position Calculation

Receiver position is computed using at least 4 satellites. The procedure is done in 6 steps:

1. Compute time of transmission:

$$T_{tr} = t - pseudorange/c \tag{E.2}$$

where

t - receiver time
$pseudorange$ - range from satellite to receiver (meters)
c - speed of light

In a case of EGNOS corrections receiver position correction will be added into pseudorange.

2. Correct the time received from the satellite (satellite clock error correction). The correction is formulated in the following way [11]:

$$\Delta \mathrm{t_{sv}} = a_{f0} + a_{f1}(T_{tr} - t_{oc}) + a_{f2}(T_{tr} - t_{oc})^2 + \Delta t_{rel} - T_{GD}{}^{1} \tag{E.3}$$

where Δt_{rel} is the relativistic correction term and can be expressed as:

$$\Delta t_{rel} = F \cdot e \cdot \sqrt{A} \cdot sinE \tag{E.4}$$

where
F - a constant $-4.442807633 \cdot 10^{-10} sec/(meter)^{1/2}$
E - eccentric anomaly (computed from the mean anomaly through iteration [refer to E.1])

Finally, the corrected time of transmission is:

$$T_{tr} = T_{tr} - \Delta \mathrm{t_{sv}} \tag{E.5}$$

[1] For L1 frequency users only
[1] $(T_{tr} - t_{oc})$should be corrected for beginning or end of week crossovers

3. Calculate satellite positions at the time of signal transmission (refer to E.1).

4. Correct satellite position by adding EGNOS correction vector:

$$SatPos^{j}_{EGNOScorrected} = \begin{bmatrix} X_{sv} \\ Y_{sv} \\ Z_{sv} \end{bmatrix} + \begin{bmatrix} \Delta X_{sv} \\ \Delta Y_{sv} \\ \Delta Z_{sv} \end{bmatrix} \tag{E.6}$$

Where

$SatPos^{j}_{EGNOScorrected}$	-	corrected satellite position
$[X_{sv}, Y_{sv}, Z_{sv}]^T$	-	raw satellite position
$[\Delta X_{sv}, \Delta Y_{sv}, \Delta Z_{sv}]^T$	-	satellite position correction from EGNOS

5. Correct satellite position for the earth rotation during the time it takes the signal to propagate from the satellite to the receiver. This is computed in the following way:

$$SatPos^{j}_{final} = R_j \cdot SatPos^{j}_{EGNOScorrected} \tag{E.7}$$

where

$SatPos^{j}_{EGNOScorrected}$	-	EGNOS corrected satellite position
R_j	-	rotation matrix
j	-	satellite number

The rotation matrix is computed as follows:

$$R_j = \begin{bmatrix} \cos(\dot{\Omega}_{\mathrm{e}} \cdot \tau_{\mathrm{j}}) & \sin(\dot{\Omega}_{\mathrm{e}} \cdot \tau_{\mathrm{j}}) & 0 \\ -\sin(\dot{\Omega}_{\mathrm{e}} \cdot \tau_{\mathrm{j}}) & \cos(\dot{\Omega}_{\mathrm{e}} \cdot \tau_{\mathrm{j}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{E.8}$$

where τ_j is signal travel time from satellite j to receiver.

6. Calculate receiver position. The position as three dimensional solution (x, y, z) is computed from pseudoranges. If the satellite positions are known, then the pseudoranges can be expressed as (solution with 4 satellites) [9]:

$$\rho_1 = \sqrt{(x_1 - x_u)^2 + (y_1 - y_u)^2 + (z_1 - z_u)^2} + c \cdot t_u \tag{E.9}$$

$$\rho_2 = \sqrt{(x_2 - x_u)^2 + (y_2 - y_u)^2 + (z_2 - z_u)^2} + c \cdot t_u \tag{E.10}$$

$$\rho_3 = \sqrt{(x_3 - x_u)^2 + (y_3 - y_u)^2 + (z_3 - z_u)^2} + c \cdot t_u \quad \text{(E.11)}$$

$$\rho_4 = \sqrt{(x_4 - x_u)^2 + (y_4 - y_u)^2 + (z_4 - z_u)^2} + c \cdot t_u \quad \text{(E.12)}$$

where

ρ_j	-	pseudorange measurements for j-th satellite
x_j, y_j, z_j	-	j-th satellite's position
x_u, y_u, z_u	-	receiver position
c	-	speed of light
t_u	-	receiver time offset

The equations above are non linear and, thus, have to be linearized.
The pseudorange can be expressed as a function of x_u, y_u, z_u and $c \cdot t_u$:

$$\begin{aligned} \rho_j &= \sqrt{(x_j - x_u)^2 + (y_j - y_u)^2 + (z_j - z_u)^2} + c \cdot t_u \\ &= f(x_u, y_u, z_u, t_u) \end{aligned} \quad \text{(E.13)}$$

Then, if we know approximate receiver position ($\hat{x}_u$, $\hat{y}_u$, $\hat{z}_u$) and time offset $\hat{t}_u$, we can rewrite previous equations as:

$$\begin{aligned} \hat{\rho}_j &= \sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2} + c \cdot \hat{t}_u \\ &= f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right) \end{aligned} \quad \text{(E.14)}$$

If we have the approximate components, then we can find true ones by adding the incremental component:

$$\begin{aligned} x_u &= \hat{x}_u + \Delta x_u \\ y_u &= \hat{y}_u + \Delta y_u \\ z_u &= \hat{z}_u + \Delta z_u \\ t_u &= \hat{t}_u + \Delta t_u \end{aligned} \quad \text{(E.15)}$$

Finally, the result can be written as:

$$f(x_u, y_u, z_u, t_u) = f\left(\hat{x}_u + \Delta x_u, \hat{y}_u + \Delta y_u, \hat{z}_u + \Delta z_u, \hat{t}_u + \Delta t_u\right) \quad \text{(E.16)}$$

The function above can be expanded about the approximate point and associated predicted receiver clock offset ($\hat{x}_u$, $\hat{y}_u$, $\hat{z}_u$, $\hat{t}_u$) using a Taylor series:

$$\begin{aligned} f\left(\hat{x}_u + \Delta x_u, \hat{y}_u + \Delta y_u, \hat{z}_u + \Delta z_u, \hat{t}_u + \Delta t_u\right) = f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right) + \\ \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial \hat{x}_u}\Delta x_u + \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial \hat{y}_u}\Delta y_u + \\ \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial \hat{z}_u}\Delta z_u + \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial_{\hat{t}_u}}\Delta t_u + \ldots \end{aligned} \tag{E.17}$$

After expansion truncation by eliminating nonlinear terms we get:

$$\begin{aligned} \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial \hat{x}_u} &= -\frac{x_j - \hat{x}_u}{range_j} \\ \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial \hat{y}_u} &= -\frac{y_j - \hat{y}_u}{range_j} \\ \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial \hat{z}_u} &= -\frac{z_j - \hat{z}_u}{range_j} \\ \frac{\partial f\left(\hat{x}_u, \hat{y}_u, \hat{z}_u, \hat{t}_u\right)}{\partial \hat{t}_u} &= c \end{aligned} \tag{E.18}$$

where $range_j = \sqrt{(x_j - \hat{x}_u)^2 + (y_j - \hat{y}_u)^2 + (z_j - \hat{z}_u)^2}$

Substituting Equations (E.14) and (E.18) into (E.17) yields:

$$\rho_j = \hat{\rho}_j - \frac{x_j - \hat{x}_u}{range_j}\Delta x_u - \frac{y_j - \hat{y}_u}{range_j}\Delta y_u - \frac{z_j - \hat{z}_u}{range_j}\Delta z_u + c \cdot \Delta t_u \tag{E.19}$$

$$\hat{\rho}_j - \rho_j = \frac{x_j - \hat{x}_u}{range_j}\Delta x_u + \frac{y_j - \hat{y}_u}{range_j}\Delta y_u + \frac{z_j - \hat{z}_u}{range_j}\Delta z_u - c \cdot \Delta t_u \tag{E.20}$$

The matrix form of the Equation (E.20) written for 4 satellites is:

$$\begin{bmatrix} \hat{\rho}_1 - \rho_1 \\ \hat{\rho}_2 - \rho_2 \\ \hat{\rho}_3 - \rho_3 \\ \hat{\rho}_4 - \rho_4 \end{bmatrix} = \begin{bmatrix} \frac{x_1 - \hat{x}_u}{range_1} & \frac{y_1 - \hat{y}_u}{range_1} & \frac{z_1 - \hat{z}_u}{range_1} & 1 \\ \frac{x_2 - \hat{x}_u}{range_2} & \frac{y_2 - \hat{y}_u}{range_2} & \frac{z_2 - \hat{z}_u}{range_2} & 1 \\ \frac{x_3 - \hat{x}_u}{range_3} & \frac{y_3 - \hat{y}_u}{range_3} & \frac{z_3 - \hat{z}_u}{range_3} & 1 \\ \frac{x_4 - \hat{x}_u}{range_4} & \frac{y_4 - \hat{y}_u}{range_4} & \frac{z_4 - \hat{z}_u}{range_4} & 1 \end{bmatrix} \cdot \begin{bmatrix} \Delta x_u \\ \Delta y_u \\ \Delta z_u \\ -c \cdot \Delta t_u \end{bmatrix} \tag{E.21}$$

The solution of this equation will give us increment components (Δx_u, Δy_u, Δz_u and Δt_u), which are used to estimate final user position and time as given in Equation (E.15) (the process requires few iterations before the correct values are reached).

Bibliography

[1] "`estb`," http://www.esa.int/estb.

[2] J. R.Kirjner, A.Lyon, "Beyond the egnos system test bed providing egnos services," ION GPS/GNSS, Portland, Tech. Rep., 2003.

[3] R. D.MANANDHAR, "Software-based gps receiver a research and simulation tool for global navigation satellite system," Center for Spatial Information Science, The University of Tokyo, Tech. Rep., 2003.

[4] P. P. Parkinson B., J.Spilker, *Global Positioning System: Theory and Applications Volume I.* American Institute of Aeronautics and Astronautics, Inc., 1996.

[5] K. B. G.Strang, *Linear Algebra, Geodesy and GPS.* Wellesley-Cambridge Press, 1997.

[6] *RTCA/DO-229C: Minimum Operational Performance Standards For Global Positioning System/Wide Area Augmentation System Airborne Equipment*, RTCA, 2001. [Online]. Available: www.rtca.org

[7] A. P.J.G. Teunissen, *GPS for Geodesy.* Springer-Verlag Berlin Heidelberg, 1998.

[8] L. Dong, "If gps signal simulator development and verification," The University of Calgary, Tech. Rep., 1999.

[9] E.Kaplan, *Understanding GPS, Principles and Applications.* Artech Housse, 2000.

[10] S. LIN, *An Introduction to ERROR-CORRECTING CODES.* University of Hawaii, 1970.

[11] J. B.-Y. Tsui, *Fundamentals of Global Positioning System Receivers: A Software Approach.* WILEY, 2000.

[12] A. M.S.Grewal, L.R.Weill, *Global Positioning Systems, Inertial Navigation, and Integration.* John Willey and Sons, Inc., 2001.

[13] Z. J.A.Starzyk, "Averaging correlation for c/a code acquisition and tracking in frequency domain," Ohio University, Tech. Rep.

[14] A. A. Alaqeeli, "Global positioning system signal acquisition and tracking using field programmable gate arrays," College of Engineering and Technology Ohio University, Tech. Rep., 2002.

[15] C. Yang, "Tracking of gps code phase and carrier frequency in the frequency domain," *IONGPS*, 2003.

[16] P. Geremia, "Cyclic redundancy check computation: An implementation using the tms320c54x," Texas Instruments, Tech. Rep., 1999.

[17] "Pegasus technical notes on sbas," EUROCONTROL, Tech. Rep., 2003.

[18] R. B. Jiming Guo, "A new tropospheric propagation delay mapping function for elevation angles down to 2°," *ION GPS/GNSS*, 2003.

[19] "Nga/nasa egm96, n=m=360 earth gravity model," http://earth-info.nga.mil/GandG/wgsegm/egm96.html.

[20] M. C.Ma, G.Lachapelle, "Implementation of a software gps receiver," *IONGNSS*, 2004.

[21] P. K.Krumvieda, DFC, "A complete if software gps receiver: A tutorial about the details," CCAR, Tech. Rep.

[22] R. B.Peeters, Univ. of New Brunswick, "Determination of the accuracy of the global positioning system s broadcast orbit and the waas-corrected orbit," Delft Univ. of Tech., Tech. Rep., 1999.

[23] S. J. Byungwoon Park, Jeonghan Kim and R. K. Changdon Kee, "The need for range rate corrections in dgps correction messages," Seoul National University and Trimble Navigation, Tech. Rep., 2004.

List of Abbreviations

ADC	Analog-to-digital Converter
AGC	Automatic Gain Control
AIAA	American Institute of Aeronautics and Astronautics
ASIC	Application-Specific Integrated Circuit
ASQF	Application Specific Qualification Facility
BPSK	Binary phase shift-keying
C/A	Coarse/Acquisition-Code
CDMA	Code Division Multiple Access
CEP	Circular Error Probable
CRC	Cyclic Redundancy Check
DGPS	Differential GPS
DLL	Delay Lock Loop
DSP	Digital Signal Processing
ECEF	Earth Center Earth Fixed
EGM-96	Earth Gravity Model 1996
EGNOS	European Geostationary Navigation Overlay Service
ESA	European Space Agency
ESTB	EGNOS System Test Bed
FEC	Forward Error Correcting

FFT	Fast Fourier Transform
FLL	Frequency Lock Loop
FPGA	Field-Programmable Gate Array
GBAS	Ground-based Augmentation Systems
GIVEI	Grid Ionospheric Vertical Error Indicator
GLONASS	Global Orbiting Navigation Satellite System
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HDOP	Horizontal Dilution of Precision
IFFT	Inverse Fast Fourier Transform
IGP	Ionospheric Grid Point
IOD	Issue of Data
IODE	IOD Ephemeris
IODF	IOD Fast Correction
IODI	IOD Ionospheric Grid Point Mask
IODP	IOD PRN mask
ION	The Institute of Navigation
IPP	Ionospheric pierce point
LSB	Least Significant bit
MCC	Mission Control center
ML	Maximum Likelihood
MOPS	Minimum Operational Performance Standards
MSB	Most Significant Bit
MSL	Mean Sea Level
NLES	Navigation Land Earth Station

PACF Access Check out Facility

PLL Phase Lock Loop

PRC Pseudorange correction

PRN Pseudo Random Noise

RF Radio Frequency

RIMS Reference and Integrity Monitoring Station

RRC Range-rate Correction

RTCA The Radio Technical Commission for Aeronautics

SBAS Satellite Based Augmentation System

SDR Software Defined Radio

SGR Software GPS Receiver

SIS Signal In Space

SPS Standard Positioning Service

UDREI User Differential Range Error Indicator

UTC Universal Time Coordinated

VDOP Vertical Dilution of Precision

WAAS Wide Area Augmentation System

WGS-84 World Geodetic System 1984