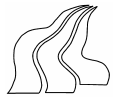
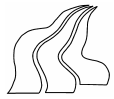
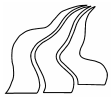






1 INTRODUCTION

1.1 BACKGROUND

Global Positioning System (GPS*) technology has become an interesting field of study for the past decade, this is because there have been significant advances made in receiver and systems technologies. These advances have enhanced the effectiveness of satellite positioning methodologies. Today, these methodologies show potential usefulness beyond their surveying roots.

Moreover, processing techniques of GPS carrier signals has been greatly improved. Attempt in finding better and better ways of increasing the stability and accuracy of the carrier signal, has led to the opening up of new possibilities with improved GPS performance not originally envisioned.

Professionals in the land surveying division can remember the hardship in establishing Geodetic Reference Frames some years ago. It was with great joy when in the early 1990s Global Position System made its debut in geodetic network densification. GPS has now become a tool for practicing surveyors and would definitely have a lasting impact on surveyors, their techniques and value of their records (www.gpsworld.com).

The technology would have many civilian users, most of whom do not belong to the survey profession. Among the civilian areas where GPS can be useful include:

- Navigation / Transportation
- Hiking
- Precision Farming
- Emergency Location
- Surveying

Today, you could be flying in a new Boeing 747 plane that displays the geographical position of the plane and its relation to nearby cities to passengers throughout the flight on a video screen in the passenger cabin. You could also rent a car that is equipped with a navigator guiding you to your hotel, displaying your position on a map and verbally giving you direction.

Presently, the technology is being applied in many fields. Some of which are:

- Intelligent / Autonomous Transportation
- Farming
- Tracking
- Surveillance

In the future, GPS technology will also be used in location based service that could include applications such as driving direction, traffic services, and and many more (www.gpsworld.com).

However, like any other technology, the system can not be said to be accurate and reliable all times. And so, to use the power of GPS fully and take advantage of the benefits it provides, we must gain a broad-based knowledge of GPS technology and its limitations. This must be seen in connection with development in desktop computing power, and innovative software approaches.

The aforementioned discussions explain why a study into the algorithms used in monitoring the accuracy and reliability of GPS technology is important.

1.2 GOAL

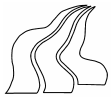
The primary purpose in this semester is to gain a broad-based knowledge into GPS observations and algorithms for processing them. The secondary goal is to perform an analysis of GPS receiver hardware with respect to its functionality and capabilities.

1.3 PRE ANALYSIS

The group was given the opportunity to work with the new Ashtech Z-Xtreme GPS receiver (ZX). This is the next generation of receiver hardware from Ashtech, replacing the Ashtech Z-12. The group embraced this opportunity, and hereby made an analysis of this receiver part of the goal for the semester.

Figure 1 Ashtech Z12 (left) and Z-Xtreme (right).

When requested, the receiver outputs a set of measurements at any given time. From these measurements the receiver position can be computed in various ways, and when compared to a known position the accuracy can be computed. For a single point position derived from the pseudorange* observations alone, the horizontal accuracy is expected to be within 100 meters (Tiberius 2000, s. 2). After selected availability (S/A*) have been turned off, the accuracy has been improved by at least a factor 10. This accuracy varies as a function of factors like numbers of satellites, multipath, delays in the ionosphere and troposphere, orbit errors, etc.



The group finds the problem with the accuracy and reliability of coordinates interesting given our surveying backgrounds, and expect that the variation of the coordinate accuracy could be analysed with some sort of satellite monitoring system.

Monitoring the available GPS satellites, by receiving and computing observations and ephemerides*, gives many opportunities for different scientific analyses. For example, the orbit of the satellites could be computed and visualised. The intension with our specific monitoring system will be to analyse, compute and visualise any important factors which has an influence on the "coordinate accuracy".

1.4 PROBLEM FORMULATION

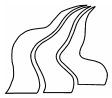
In realizing the above mentioned goal and pre analysis, and given the fact that the group was given the opportunity to work with the new Ashtech Z-EXtreme GPS receiver, the following problems/questions have been defined.

- How to establish a communication protocol with the receiver with an appropriate and suitable software/program?
- Explore the data types and data formats available in the receiver, and choose between these, the format for measurement and ephemeris data.
- How this raw data should be handled, to make it ready for further computations?
- Which computations are needed, to establish receiver position, and which should be made for visualization purposes?
- How do we present these visualizations graphically?

These rather open questions have been re-defined into theoretical problems, which will be the guide throughout the report.

- How is it possible to create a GPS-monitor and interface to the ZX, with the purpose of analysing the coordinate accuracy?
- What would be the most interesting features to include in this monitor/ interface, to obtain knowledge about the factors that influence the coordinate accuracy?

The theoretical approach to the solution of these problems is described in the next chapter, where analysis of the different problems is made.



2 ANALYSIS

The purpose with this chapter is to discuss the given problems from the problem formulation. Some problems, and their solution needs a theoretical background, while for others the chosen solution will only be explained. The analysis will complete the full background for the system, prior to the actual development.

2.1 MONITOR FEATURES

In this chapter it will be discussed what to include in a software with the purpose of monitoring GPS satellites in real-time, with the position accuracy as main interest area.

After having discussed the problem in the group and with our professor, it was decided to include the following elements in the GPS monitor. The purpose and reason for each element will be discussed below.

Receiver position

The real-time receiver position and some form of calculations on variance and standard deviation must be computed and visualised. Since the monitor will run for an indefinite period of time, the computations on standard deviations would have to be time dependent, to show a more actual variance. For example it could be calculated with 10 previous measurements.

Stereographic plot of satellites orbit

The receiver position accuracy is highly dependent of the satellites positions in space, and a good spread around the horizon is crucial for accurate computations. A skyplot which gives a panoramic visualisation of the satellites real-time positions in space would give a good view of the given situation.

The quality of a given constellation of satellites can also be defined by values, which will be described below.

Important values

In GPS technology various factors have significant influence on the coordinate accuracy. These values are therefore also important elements in the monitor.

DOP values

The above mentioned constellation of satellites in space is described by DOP* values. The DOP values are given for the total geometry GDOP. From this four others can be derived. The PDOP, HDOP, VDOP and TDOP. The theory for these are all described later on.

Health

All satellites broadcast their current health by the ephemerides. There is one value for good health and another one for bad when the satellite is unable to operate within the specified tolerance. Normally, all available satellites indicate good health, but this can not be taken for granted. The group have during the experiments at one point encountered 2 out of 6 satellites with bad health indications, with a poor coordinate accuracy as a result. This feature is therefore also included in the monitor. Further more, a check on the satellites health, ensures that computations only occur with 5 or more satellites indicating good health.

Signal to Noise Ratio indicator

The Signal to Noise Ratio (SNR) is the ratio of desired signal to unwanted noise. A high signal to noise ratio is highly desired. Difference of 45 dB is required for clear reception. It varies from approximately 30 up to approximately 55 Db*Hz. The purpose with this feature is to visualise the quality of the signal received from the tracked satellites.

The plot of important values will therefore include DOP values, health and the signal to noise ratio. The computation of these values also includes the number of satellites with good health, which will be checked before further computations are engaged.

Sub Satellite Point

The primary objective of the earlier mentioned stereographic plot, is viewing the satellite orbits from the receivers point of view. Another way of visualising the current satellites orbits would be to see the full orbit picture on a world map. This would give a clear indication, of where the individual satellites are located in space above the earth.

The orbits are drawn as a function of their vertical position above the surface of the earth, at any given time. To ensure the best comprehension of such a visualisation, a widely understood map projection must be chosen. The method is further explained later on.

Available satellites for the day

Finally a chart of the available satellites for the current day can be calculated and visualised. The primary reason for such information, is the possibility to check for the number of available satellites above a given elevation mask*, at any given time of the day. This enables an easy job planning procedure, and bad periods can be excluded for surveying purposes.

2.2 PRESENTATION

The monitor can be presented in various ways. The preferable way, would be to make an interface to the main program using a GUI (graphic user interface). All the mentioned monitor functionalities could also be handled a more simple way by using standard plot functions. The intension is to establish all wanted features as plots to begin with and then later on try to incorporate them in a GUI, if time allows it.

2.3 SERIAL COMMUNICATION

Communication with the receiver was done through an RS-232 serial interface. This section discusses general overview of the RS-232 serial port.

2.3.1 Data Format

Data format in RS-232 serial ports includes 1 start bit, between 5 and 8 data bits, and 1 stop bit. A parity bit and additional stop bit may be included.

The diagram below illustrates serial data format.

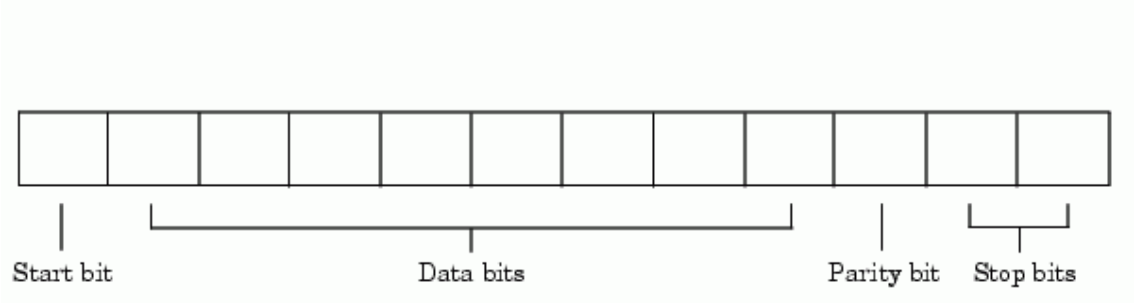
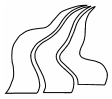


Figure 2 Data format

Normally serial port data is often express as:

- Data Bit
- Parity
- Stop Bit

8-N-1 represents 8 data bits, no parity, and 1 stop bit. Usually, 8 data bit are always used in storing serial data in a computer regardless of the number of data bit specified. In reading or writing data, a value is always specified. For example when one value from the receiver is read with the int16 format, then that value consists of 2 bytes.



2.3.2 Data Bits

Devices such as GPS receiver transfer data through the serial port as data bit, and can either be ASCII (text) data or binary data. These data bits transferred may be receiver commands, readings or error messages, etc. Binary data are transmitted as 8 bits. ASCII data is transmitted as either 7 or 8 bits (Mathworks).

2.3.3 Communication Protocols

Two standard types of protocol are usually supported by the RS-232 serial port. They are:

- Synchronous
- Asynchronous

In synchronous protocol as the name suggests, all data transmitted are synchronized to a common clock, whereas in asynchronous protocol each device uses its own internal clock. The former can be said to be faster than the latter since bits to mark the beginning and end of each data byte are not required. That is to say, in synchronous protocol, each bit that is transmitted is either an actual data or an idle character.

However, most serial ports use the asynchronous protocol because it has the advantage that the processor does not have to deal with the idle character (Mathworks).

2.4 RECEIVER POSITION

According to lecture notes from the GPS Fundamentals and Algorithms course in February, 2002, the following can be said.

Code observation equation that excludes ionospheric delay, troposheric delay and satellite clock error expressed in meters is given by:

$$p_r^s = \rho_r^s + cdt_r \tag{2.1}$$

Where ρ_r^s - is the range between satellite and receiver, also called pseudo range.

dt_r - is the receiver clock offset

c - Velocity of light in vacuum.

Also

$$\rho_r^s = \left\| x^s(t-\tau) - x_r(t) \right\|$$

(2.2)

Where t - time of observation at receiver

τ - travel time

For a single point positioning that includes four code observations, equation (2.1) can be expressed as

$$\begin{bmatrix} p_r^1 \\ p_r^2 \\ p_r^3 \\ p_r^4 \end{bmatrix} = \begin{bmatrix} \rho_r^1 + cdt_r \\ \rho_r^2 + cdt_r \\ \rho_r^3 + cdt_r \\ \rho_r^4 + cdt_r \end{bmatrix}$$

(2.3)

This gives four observation equation with four unknown parameters, x_r, y_r, z_r and receiver clock error dt_r.

The code observation from equation (2.1) could also be defined as

$$p_r^s = \sqrt{\left(X^s - X_r\right)^2 + \left(Y^s - Y_r\right)^2 + \left(Z^s - Z_r\right)^2} + cdt_r .$$

(2.4)

Linearizing the above equation for several observations, we have:

$$p_r^s - \rho_{r,o}^s = -\frac{X^s - X_r^o}{\rho_{r,o}^s} x_r - \frac{Y^s - Y_r^o}{\rho_{r,o}^s} y_r - \frac{Z^s - Z_r^o}{\rho_{r,o}^s} z_r + cdt_r$$

(2.5)

(X_r^o, Y_r^o, Z_r^o) are approximate/preliminary receiver coordinates. Satellite coordinates are known.

$\rho_{r,o}^s$ is approximate distance between satellite and receiver, computed from approximate receiver coordinates (X_r^o, Y_r^o, Z_r^o).

Rearranging equation (2.5) in a matrices form for m observations, we have,

$$\begin{bmatrix} \Delta p_r^1 \\ \Delta p_r^2 \\ \vdots \\ \Delta p_r^m \end{bmatrix} = \begin{bmatrix} -\frac{X^1 - X_r}{\rho_r^1} & -\frac{Y^s - Y_s}{\rho_r^1} & -\frac{Z^s - Z_r}{\rho_r^s} & 1 \\ -\frac{X^2 - X_r}{\rho_r^2} & -\frac{Y^2 - Y_s}{\rho_r^2} & -\frac{Z^2 - Z_r}{\rho_r^2} & 1 \\ \vdots & & & \\ -\frac{X^m - X_r}{\rho_r^m} & -\frac{Y^m - Y_s}{\rho_r^m} & -\frac{Z^m - Z_r}{\rho_r^m} & 1 \end{bmatrix} \begin{bmatrix} x_r \\ y_r \\ z_r \\ cdt_r \end{bmatrix}$$

(2.6)

Δp_r^s is the observed minus computed observation.

We now have an observation equation of the form

$$b = Ax$$

(2.7)

Assuming an independent code observation with equal variance, the least square solution is given by,

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ \hat{cdt}_r \end{bmatrix} = \left(A^T A\right)^{-1} A^T b$$

(2.8)

It can be stated here that, the algorithm used in computing the receiver position puts the initial receiver position and initial clock offset to zero. Hence the residuals obtained from the least square solution for the first iteration gives a value that is close to the true values.

The estimated receiver coordinates is then given by

$$\hat{X} = X_r^o + \hat{x},$$
$$\hat{Y} = Y_r^o + \hat{y},$$
$$\hat{Z} = Z_r^o + \hat{z},$$

(2.9)

These estimated values are then used as new approximate receiver position and clock offset and then used for the next iteration. Several iterations are done until an acceptable solution is obtained.

In the programme that computes the receiver's position in this project, this procedure is repeated for every epoch.

2.5 DILUTION OF PRECISION

The covariance matrix obtained from computing $\left(A^T A\right)^{-1}$ gives information about the geometric quality of the receiver position. Normally, when the satellites used in computing the receiver position are well oriented (evenly spaced), dilution of precision is smaller and the computation is more accurate (Strang & Borre 97).

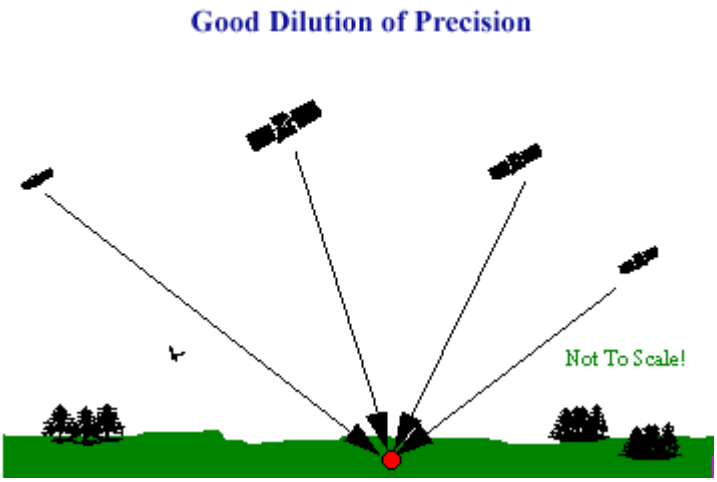


Figure 3 Good Dilution of Precision

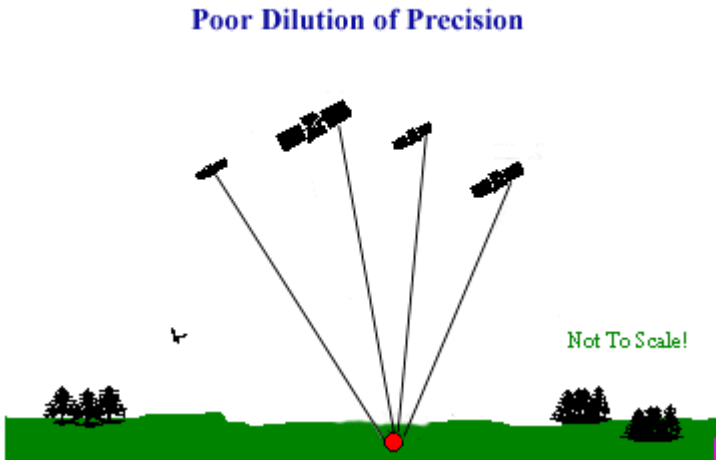


Figure 4 Bad Dilution of Precision

DOP values are dependent on the satellites position in space. Since the satellites are constantly in motion the DOP values are always changing.

Derivation of dilution of precision (DOP) starts with the covariance matrix.

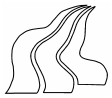
$$\left(A^T A\right)^{-1} = \begin{bmatrix} \sigma_X^2 & \sigma_{XY} & \sigma_{XZ} & \sigma_{Xcdt} \\ \sigma_{YX} & \sigma_Y^2 & \sigma_{YZ} & \sigma_{Ycdt} \\ \sigma_{ZX} & \sigma_{ZY} & \sigma_Z^2 & \sigma_{Zcdt} \\ \sigma_{cdtX} & \sigma_{cdtY} & \sigma_{cdtZ} & \sigma_{cdt}^2 \end{bmatrix} \tag{2.10}$$

Geometric Dilution of Position (GDOP) is given by:

$$\text{GDOP} = \sqrt{\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2 + \sigma_{cdt}^2} \tag{2.11}$$

Other dilution of precision (DOP) values that are in use to determine the precision of various component of position and time computation are:

- Position (PDOP) = $\sqrt{\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2}$
- Horizontal (HDOP) = $\sqrt{\sigma_X^2 + \sigma_Y^2}$



- Vertical (VDOP) = $\sqrt{\sigma_Y^2} = \sigma_Y$

- Time (TDOP) = $\sqrt{\sigma_{cdt}^2}$ /c

It is known from experience that a good observation is achieved when PDOP is less than 5, and measurement is made from 5 or more satellites (Strang & Borre 97).

2.6 GPS TIME

Precise knowledge of time is a key factor to accuracy in GPS measurement. GPS message contains information that allows a receiver to convert GPS Time into Universal Time Coordinated. Beginning from midnight of Saturday (05/01/1980) to Sunday (06/01/1980), GPS Time started counting uniformly in weeks and seconds of week. The weeks always begin at the Saturday/Sunday transition.

GPS week 0 began at the start of the GPS time scale. The time within each week is usually denoted as the seconds of week (sow), and is between 0-604800 (60 x 60 x 24 x 7).

From the origin of GPS time, only 1,024 weeks were allotted before the system reset to zero because 10 bits are allocated for the calendar function (2^{10} is 1024). Thus the 1st GPS rollover occured at midnight on August 21, 1999. The next GPS rollover will take place May 25, 2019.

The GPS time scale is synchronised to keep steps with Universal Time Coordinated (UTC) and International Atomic Time (TAI). However, UTC differs from TAI by an integral number of leap seconds to maintain correspondence with the rotation of the earth, whereas GPS time does not include leap seconds. At present TAI is ahead of UTC by 32 seconds, and ahead of GPS time by 19 seconds. And GPS is ahead of UTC by 13 seconds.

Almost all GPS computations need time tag in seconds of week. This can usually be derived from the receiver. The Astech Z-Xtreme receiver gives epoch time in units of 50ms, modolu 30 minutes. And the range is 0-36000. It is therefore necessary to convert this time format to GPS time. Moreover, GPS calculations on millimeter level require that time is kept to nanoseconds. Hence, there is therefore a need for a correction of time so as to avoid underflow or overflow. The matlab function that does this is check_t. This function sets GPS seconds of week in the range ± 302400 as specified in the ICD200.

2.7 PSEUDORANGE

Pseudorange* is a fundamental measurement in GPS. In pseudorange measurements, the user equipment receives Pseudo Random Noise (PRN)* code from the satellite. After having identified the satellites, the receiver generates a replica code.

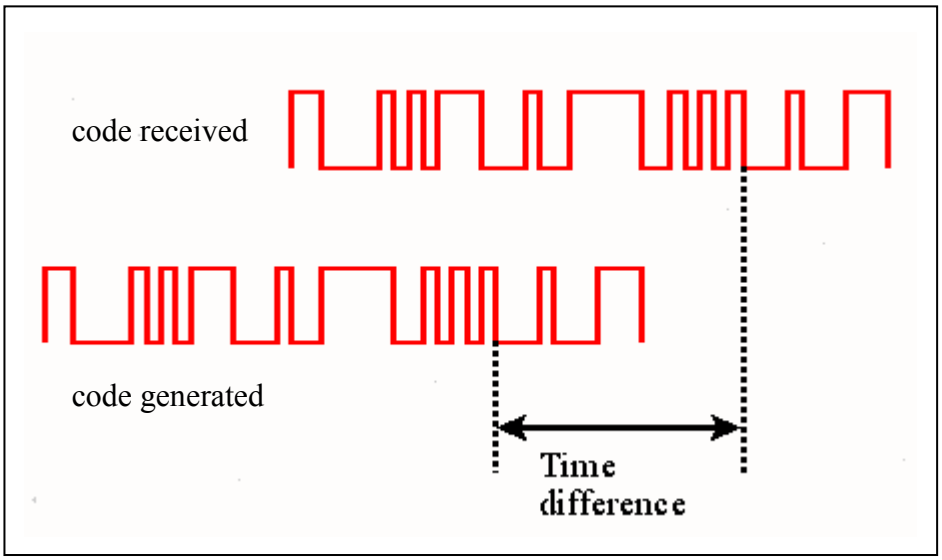


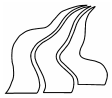
Figure 5 Pseudorange Random Noise Code

To obtain an approximation of the satellite range, the replica code in the receiver is shifted in phase to maintain a maximum correlation with the satellite code. The phase by which this code is shifted is then multiplied by the speed of light to obtain an approximation of the satellite range. It is called pseudorange because the measurement must be corrected by a variety of factors to obtain the true range.

Some of the correction factors include:

- Ionospheric delay
- Tropospheric delay.
- Clock errors.

The true geometric distance to each satellite is then obtained by applying these corrections to the measured pseudorange. Some of these error sources are discursed in the next section.



2.8 GPS ERROR SOURCES

It is an established fact that no GPS observation is perfect. GPS positions have some errors in them that affect the accuracy of the observation. Some of the errors that affect the accuracy of the observations are:

- Clock errors.
- Ionospheric delay
- Tropospheric delay.
- Orbital error.
- Multipath error.
- Receiver errors.

A number of these errors can be modelled and eliminated in GPS observations and computations. These errors include clock errors, ionospheric and topospheric delay, orbital error and receiver errors.

2.8.1 Clock Errors

Both the satellites and receivers need very good clocks to do their job. Smallest error can throw off the range measurement from the receiver to the satellite. For example a 10 nanosecond (0.00000001sec) error would cause a 3-meter error in range.

2.8.2 Ionospheric Delay

The ionosphere is the upper part of the earth's atmosphere that starts about 50km above the earth and extends to about 1000km (Teunissen $ Kleusberg 1998). It contains charged particles called ions and electrons that affect the propagation of radio waves. The amount of charged particles in the ionosphere is proportional to the radiation intensity of the sun.

GPS signals from the satellite to the receiver gets delayed as they travel through an area of charged particles. The signals are refracted as they pass through charged ions and electrons in the ionosphere and are therefore disperse. This effect is frequency dependent. The error could be up to 30m, and differs primarily for night and day.

2.8.3 Tropospheric Delay

Troposphere is the lower part of the earth's atmosphere that extends to about 50km above the earth. This part of the atmosphere contains neutral atoms and molecules.

In the troposhere signal path gets diffracted as they pass through. The effect does not depend on frequency. Error caused by tropospheric delay is also between 0-30m.

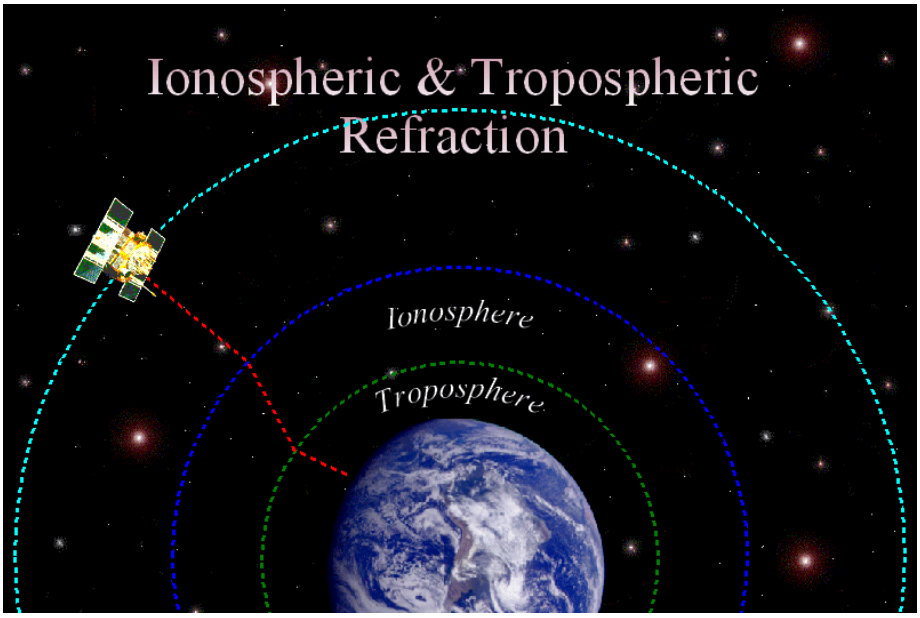


Figure 6 Ionosphereic and Tropospheric delay and refraction

Delays in both ionosphere and troposhere depends on elevation of satellite.

2.8.4 Orbital Error

The positions of the satellites obtained from the ephemerides are really a prediction of where the satellite should be at a given moment, and can differ slightly from the actual position. Necessary steps are taken to predict the best positions (or orbits), but they can not be predicted perfectly all the time. Orbital errors can be in a range of 1-5 m.

Figure 7 Satellite Orbit

2.8.5 Multipath Error

GPS signal may bounce off a nearby object. This signal arrives at the receiver: first the direct one and then a bunch of delayed reflected ones. This gives interference and can be confusing to the receiver especially if bounced signals are strong enough. The result is an erroneous measurement. Signals can be reflected from buildings or from the ground and create error of several meters on code observation. Error could be between 0-1m.

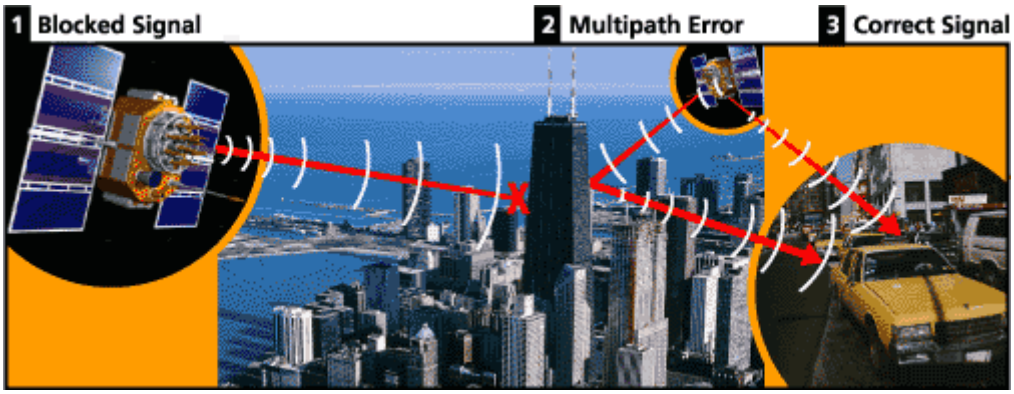
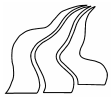


Figure 8 Multipath Error

Multipath can not easily be modelled and hence is a serious problem. Improving the site of the receiver is an attempt to minimize multipath errors. Other ways of eliminating this error is a good antenna design.



2.8.6 Receiver Errors

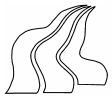
This is a function of how well a GPS receiver can measure the signal coming from the satellite. Some are better at it than others.

2.8.7 Summary of GPS Error

Listed below is the extent to which each of the above errors can affect GPS Position measurements. The chart below is commonly refered to as GPS error budget.

Source of Error	Error in meters
Clock Drift	0-1.5
Ionosphere	0-30
Troposphere	0-30
Orbital	1-5
Multipath	0-1
Measurement Noise	0-10

Figure 9 GPS Error budget



3 SPECIFICATION

After having analysed which features to include in the GPS monitor, and how to present it on the screen, it is now possible to draw out the specifications for the GPS monitor, and the different software and hardware used to compute this.

3.1 COMMUNICATION

Concerning communication, the group found it appropriate to use numerical modelling tool like Matlab. Communication protocol should also be asynchronous in order to read and write to the receiver simultaneously.

3.2 DATA PROCESSING

The following points have been identified concerning data processing:

- The data processing has to result in real-time availability of input to the monitor.
- Amount of logged data shall not exceed the storage device's capabilities.

3.3 MONITOR FEATURES

It was decided that the monitor should include the following features.

- Graphical display of receiver position/accuracy
- Stereographic plot of satellites orbit.
- DOP values.
- Satellite Health.
- Signal to noise ratio indicator.
- Plot of sub-satellite point.
- Prediction of available satellites for the day

4 SYSTEM DESCRIPTION

This chapter gives a detailed description of the various sections that make up the system.

It begins with the description of how we were able to establish a communication with the receiver, and then a description of the Matlab functions used in data handling and processing. It concludes with a discussion of the various plots.

4.1 OVERVIEW

The following section gives a general overview of the components of the system used as an implementation of the solution as stated in the problem formulation (see section1.4)

The implementation consists of a Pentium 3 PC Serial Interface, a Matlab workspace, and Ashtech Z-Xtreme GPS Receiver (ZX).

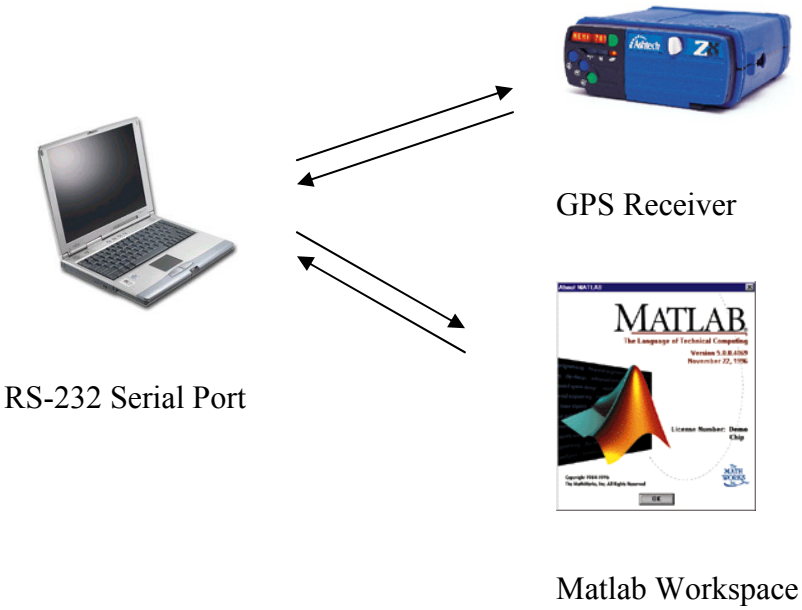
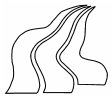


Figure 10 Overview of the setup



4.2 SOFTWARE

The following general functionality has to be provided by the developed software.

- Establish communication from Matlab workspace to the Z-Xtreme GPS Receiver.
- Handle both binary and ASCII data
- Process data in Real-Time

This functionality is described in the following under the headings: Serial Communication, Data Handling on- and off-line, and Data Processing.

The software was developed with Matlab Version 6 development environment. The group found this tool best for the intended application.

The relation of the various m-files is given in fig (10). An overview of the developed program (flow chart) can be gain from fig (11). And the source code is provided on CD.

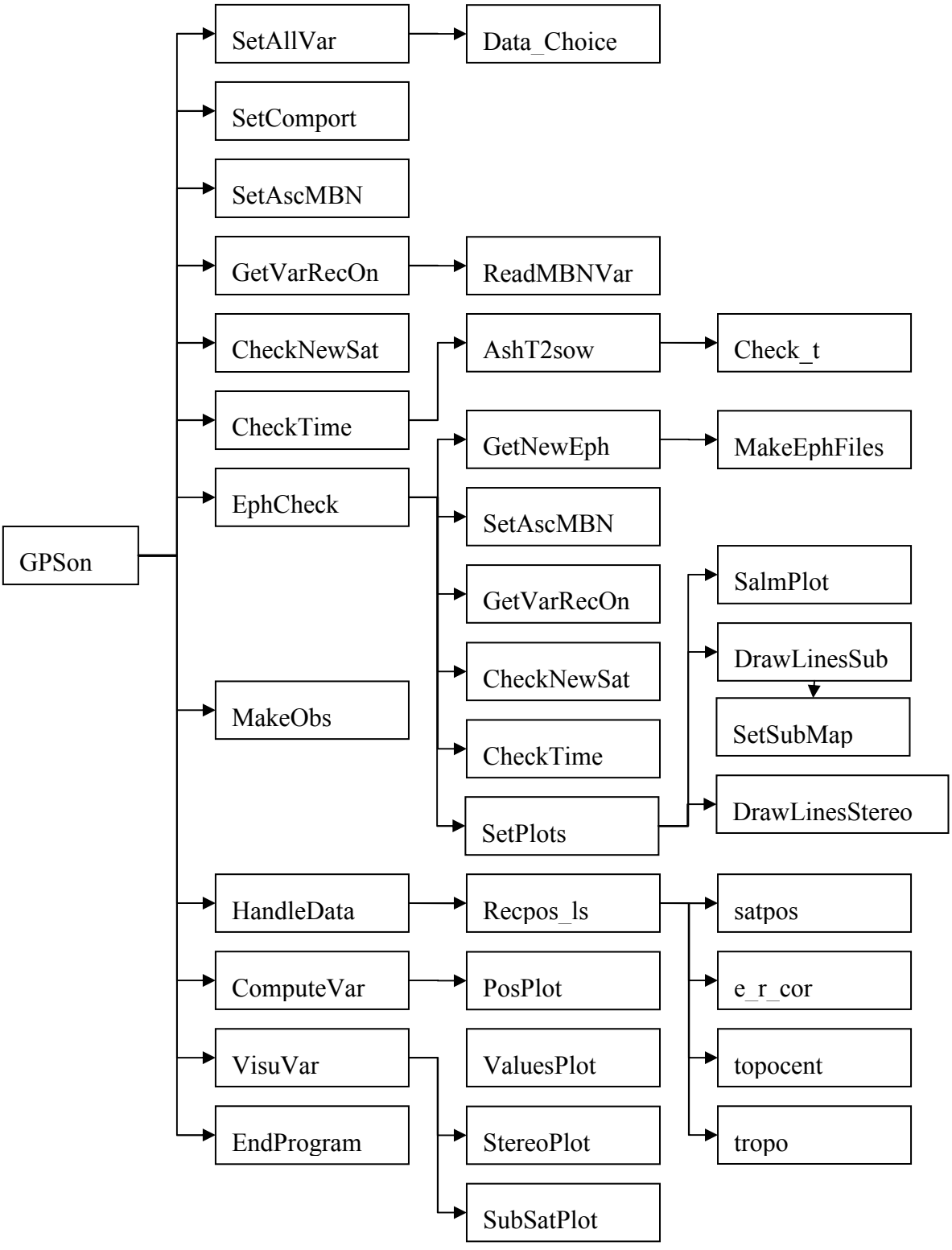


Figure 11 Relation diagram of the m-files in the system

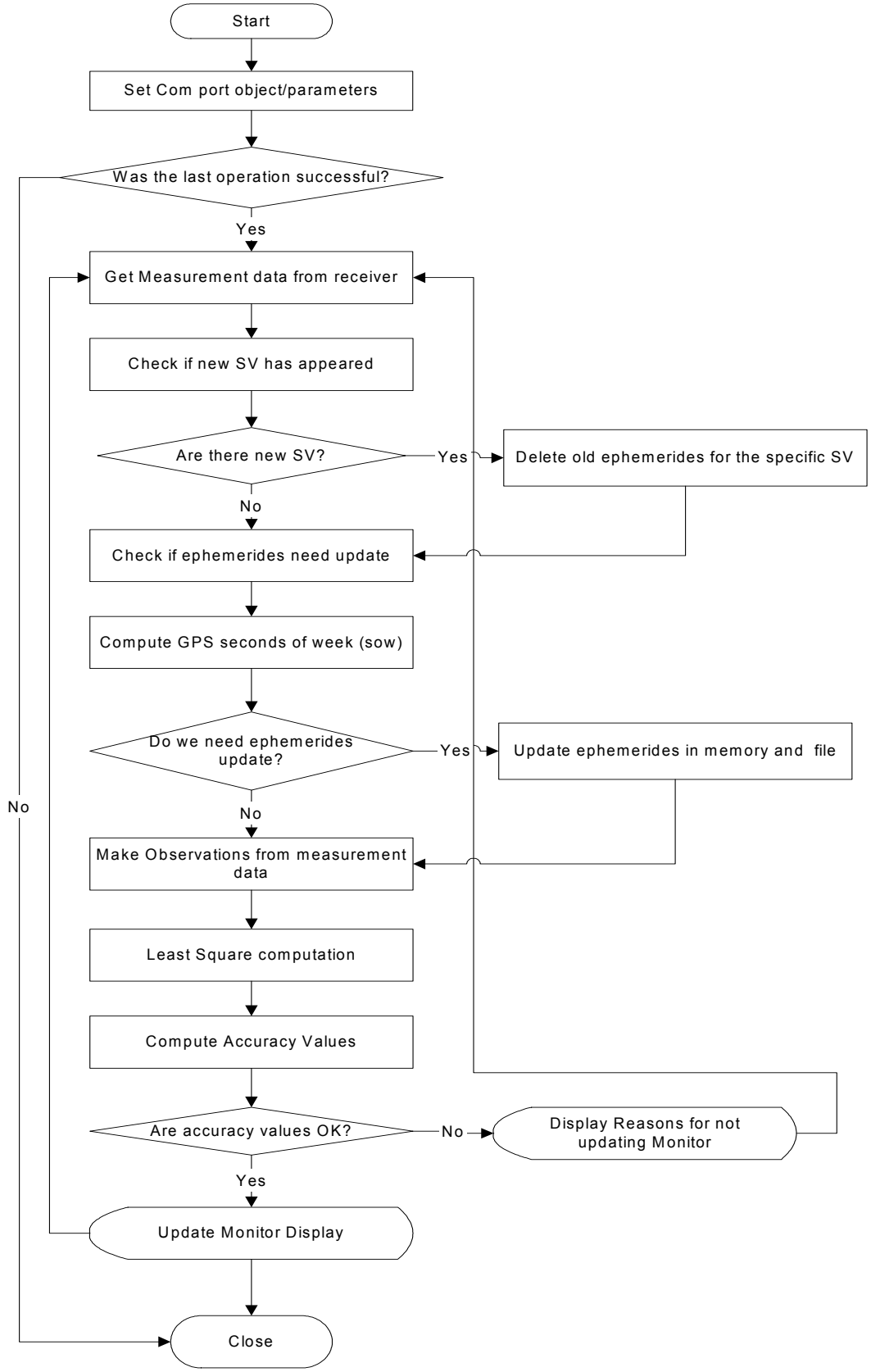
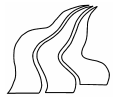


Figure 12 Flow chart of the overall program



4.3 SERIAL COMMUNICATION

Matlab was used in writing and reading data to the receiver through the serial port. In doing so the following steps were taken:

- Serial port object was created in Matlab workspace.
- Communication settings were configured.
- Data were written and read from the receiver.

The following sections explain how each of the above tasks was done.

4.3.1 Creating the Serial Port Object

In creating the serial port object in Matlab, the serial function was used. The function requires the name of the port connected to the receiver as an input argument.

The command: S = serial ('COM1')

Creates a serial port object associated with com1 port .

Once the serial port object is created, its properties are automatically assigned.

4.3.2 Configuring the Port Settings

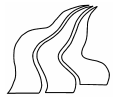
It is important that both the receiver and the serial port object have identical communication settings before we could write and read data. If they are not identical, then we could not successfully read and write data.

The settings consisted of specifying the values for the following properties.

Baud Rate*	9600
Data Bits	8
Parity	None
Stop Bits	1
Terminator	LF

Figure 13 Configuration settings

The Matlab Command set was used in setting the communication properties. The same command was used in setting the time out (this is the length of time elapsed before an error message is output if no data is transferred).



The receiver was also set to the same values.

4.3.3 Writing and Reading Data

Asynchronous write and read was used because of the following reasons:

- Not to block access to the Matlab command line.
- To enable us execute read (write) operation whilst other write (read) operation is in progress.

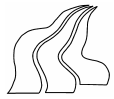
Matlab command fprintf was used to write to the receiver. Asynchronous read was done in Matlab by configuring the *ReadAsyncMode* property to continuous. This made it possible for the serial port object to continuously query the receiver to read available data. Available data is read either by an fgetl (if output is expected to be receiver command), fread (if output is binary).

It is worth noting here that before writing and reading to the receiver, the receiver must first be connected to the serial port object by a fopen command. When the serial port object is no longer needed, it was disconnected, removed from memory and Matlab workspace by the following commands.

```
fclose(s)
delete(s)
clear(s)
```

It was realised that if an error is encountered in the middle of a run of the programme, Matlab stops running and an error message is returned. But the serial port object is not closed and disconnected since the part of the programme that closes and disconnects the serial port object has not yet been reached. This makes communication with the receiver again through the port impossible unless Matlab is completely exited and re-launched.

This was solved problem with the try, catch and end functions in Matlab. This means that opening the serial port object and communication with the receiver as well as real-time data handling and processing is done between try and catch functions. If an error is encountered between a try and catch, the programme jumps into a catch and end, where the serial port object is closed and disconnected. The last error message is also displayed here. This way, communication again with the receiver becomes possible without quitting Matlab and re-launching.



4.4 DATA HANDLING ON-LINE

The handling of data primarily happens when the receiver is connected and running (on-line). All files and procedures described below will therefore concern the handling of data on-line.

4.5 DATA FORMATS IN ASHTECH Z-XTREME

The receiver can be set to give an output in either binary or ASCII format, depending on the data type. All 8 data types are available in binary, while only 3 are ASCII. The table below shows the data types, and their description.

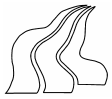
Raw Data Type	3-Character String	Description	Format
MBEN	MBN	Measurement data	ASCII / Binary
PBEN	PBN	Position data	ASCII / Binary
SNAV	SVN	Ephemeris data	Binary only
SALM	SAL	Almanac data	Binary only
EPB	SPB	Raw ephemeris	Binary only
DBEN	DBN	CPD carrier phase	Binary only
CBEN	CBN	CPD position data	ASCII / Binary

Figure 14 Raw Data Types and formats. [Technical reference manual p. 129.

Only the measurement data and the ephemerides are required to establish the monitor's functionality described in the problem formulation. The almanac data could also be included, if orbit computations of all satellites are to be made for a longer period of time. Though it is not a necessity, since almanac data also could be derived from the ephemerides, when they arrive.

4.5.1 Data query and receiver settings

After having established communication with the receiver, it is now possible to query for information, and set the receivers different parameters. At first the site name is set from default "????", to a number 9999, and the elevation mask is set to 15 degrees. Then the receiver is set to give ASCII measurement data every 5 sec. This settings is



applied by running the m-file SetAscMBN .There are many different settings available, but no further are needed at this point.

As just explained it is possible to query for either binary or ASCII data. It is important to note that this is not possible simultaneously. It is therefore necessary to query for binary and ASCII data independent of each other, and establish their own pre-settings before using the specific data query commands. To perform a switch form one format to another, different approaches are needed.

When switching from ASCII to binary format, the receiver is set to a 5 second output rate of measurement data. This is terminated by using a receiver reset command. Then the receiver is set to give a binary output, each line individually queried for.

When going back to ASCII format, the m-file SetAscMBN, is called again, and new measurement data are delivered every 5 sec for 20 epoch. Then the system is slowed down by running the m-file slowdown. Here the output rate is changed form 5 sec to a slower rate. For example 30seconds.

The handling of the binary and ASCII data is thoroughly discussed below.

4.5.2 ASCII Data

It was earlier decided to collect the measurement data in ASCII format. As explained, this is done, by running m-file SetAscMBN. This sets the receiver site name, elevation mask and output format to ASCII with an output rate of 5 seconds. Afterwards the receiver returns a number of lines with measurement data beginning with "$PASHR,MBN" equivalent to the number of tracked satellites.

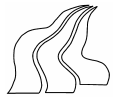
The returned lines are checked with the command strncmp(GetL,' $PASHR,MBN',10), in the m-file GetVarRecOn before they are handled in the m-file ReadMBNVar. This check may not seem very important, but it makes the program more stable, and makes it very easily to expand. If for example position data PBN also was wanted in every epoch, they could easily be added as queries in the SetAscMBN and sorted to the correct handling program in GetVarRecOn

Below is a detailed discussion of the m-file that handles the ASCII measurement data.

ReadMBNVar

Functionality

The m-file reads the 37 different variables in the MBN measurement data string for each satellite, and assign them to an 37 x (number of satellites) array. If a line is corrupted, the epoch is discarded and the procedure is re-run, with a new epoch of data. A warning massage is also displayed!



If it was chosen during the start-up, a data file called MBNVar.dat is created during the run of the program. Here the lines of measurement data are appended one after the other.

Procedure:

The m-file uses a special feature incorporated in the returned lines of measurement data, to determine the end of each epoch. The second variable in each data string indicates the number of remaining lines of measurements. (Number of remaining structures NRS)

Example of the first string in an epoch with 5 available satellites is:

$PASHR,MPC,21400,**04**,20,64,244,02,002,24,5,053,00,-02262……

Here the NRS is the highlighted 04, meaning there are 5 satellites currently tracked above the elevation mask. The remaining 4 satellites would have their NRS in the measurement as 03, 02, 01, and 00. When the NRS = 00, the measurement data for the epoch has finished. This is therefore used as a stop for the while loop, that is reading the strings, and collecting measurement data for the given epoch.

If wanted every line is stored in a file by running MakeMBNVarFile. Here the line is added to the MBNVar.dat file by using fwrite(writefid,GetL,'char').

In the while loop, the command strtok(tline,',') is used, to get the next variable, which are all separated with a comma as delimiter. The command str2num converts the collected variable to a number, and assign then to a row in a temporary 36 x 1 array M. Before the loop is repeated, the array M is assigned as the next column in the measurement data array MBNVar.

When all the measurement data for the epoch has been read, the array MBNVar will be an 36 x 5 array for the given example.

The structure of the variables follow the structure of the measurement data string, which is described in the Ashtech Technical manual p. 141-142. For the calculation of the position, only the time tags, and the C/A code observations (Code transmit time), are needed.

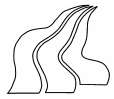
The whole procedure is within a try, catch, end loop, to catch corrupted lines of measurement data. The procedure is re-run, if one line is corrupted.

CheckNewSat

Every time an epoch of measurement has been computed into the array MBNVar, it is checked for any new satellites. This is done by running the command

```
NewSat = SETDIFF(MBNVar(3,:),MBNVarold(3,:));
```

Here the satellites PRN number in the new array is compared with the previous one.



If one or more satellites have appeared in the new MBNVar array, the array containing the ephemerides EPH is checked for matching ephemeris. If any are found their age is computed, and if they are older than 2 hours they are removed from the array. This will engage an ephemeris update, which is explained in the next chapter.

If an ephemeris update will be engaged, the ephemerides for the tracked satellites are all checked. If any are older than 2 hours, they will also be removed, and therefore updated at the same time.

The first reason for this procedure is that a satellite could disappear shortly behind an obstacle. When reappearing, there would be no need for getting a new ephemeris, since the one already collected would be good enough. Secondly checking all tracked satellites for old ephemerides, will ensure that the software always is working with the newest ephemerides, and it also reduces the number of ephemeris updates needed.

4.5.3 Binary Data

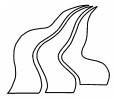
To query for binary data, the Matlab command fread(s,size*)* is used. It is therefore necessary to know the exact size of the data needed. According to the manual one line of ephemeris, containing ephemeris for one satellite is 145 bits. This includes a start and a stop bit, plus a terminator.

At start-up, there is no information about tracked satellites, or which are included in the broadcasted ephemerides. A query for ephemerides at this time would therefore impose problems, given the lack of information. It is possible to try a query for example 12 ephemeris using fread(s,1740*),* but chances for a timeout warning are great, because ephemeris for 12 satellites rarely are available.

It is therefore better to delay the query for ephemerides, till sufficient information about observed satellites is established. Then the query would be much more precise, because you could specified the exact satellites for which ephemerides are needed, by making an ephemeris check.

This procedure is equivalent to an ephemeris update, for a given number of satellites, because it compares observed satellites and known ephemerides, when checking for needed ephemeris. If no ephemerides are known, for example at start-up, this method would collect all ephemeris needed.

Since an ephemerides update routine had to be made anyway, it would be practical if this routine could handle several ephemeris updates simultaneously. Both because it would be able to get all needed ephemerides at start-up, but also because more than one satellite in theory could become visible to the receiver, within one update loop. The



ephemeris update method, has therefore been chosen, as the primarily way to collect ephemerides in the program. The m-files used in this method are described below.

EphChech

Functionality:

Checks for satellites with no assigned ephemerides within the main update loop of the program. If any found, it runs m-file GetNewEph, SalmPlot, DrawlinesSub and DrawlinesStereo. These m-files will be described later on.

Procedure:

Here the command setdiff is used to compare satellites PRN number. found in the MBNVar array, and satellites PRN nr in the ephemeris array. The command returns SatNoEph (satellites with no ephemeris), which will be the input to the m-file GetNewEph. The first run will return all available satellites, because the ephemeris array primarily is set to zeros (21,1). After the first run, only new satellites above the elevation mask are detected.

GetNewEph.m

Functionality

Collects the ephemeris for the number of satellites, specified in the SatNoEph array, and assign a specified number of variables to an 21 x (number of ephemeris) array. The 21 chosen variables follow a convention so the structure can function with other programs and m-files. Especially those made by Kai Borre which is described in (Strang & Borre 97).

Procedure:

At first this m-file resets the receiver to exit ASCII mode. Afterwards, it sets the receiver to binary output using

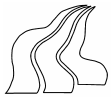
- fprintf(s,'$PASHS,RST')
- fprintf(s,'$PASHS,OUT,A,BIN')

Then the needed ephemeris are queried one after another in a handling loop, with the command

- fprintf(s,'$PASHQ,SNV,28') (e.g query for ephemeris for satellite 28)

The returned binary string, is collected with fread(s,145).

Handling this binary string was not as easy as implied by the Ashtech manual. The structure was well described, but especially doubles and floats came out as odd numbers. Apparently the string wasn't read correctly, for one reason or another. Finally Kai Borre found out, that a "b", had to be introduced, in an fopen statement when reading



the binary data from a file, to set the binary machine format to ”IEEE floating point with big-endian byte ordering1”. This is no problem when working offline, because the data is stored in a file. However the method is not suitable for working online, because here no data is stored in files.

After having worked with this problem for quite some time, the solution of writing and reading a file was made, so the project wouldn't be further delayed, by this problem.

After getting the binary string using fread, the data is stored in a file using the command

- writefid = fopen('OneEphFile.bin','w','b')
- fwrite(writefid,fid)

And then read again using

- Ephemerisfile = 'OnEphFile.bin'
- Fide = fopen(ephemerisfile,'r','b')

Now the binary data are ready to be handled, and the variables can be assigned. There are 31 variables available, but only 21 are assigned to the ephemeris array eph. The structure of this array can be seen in appendix A.

The rest of the variables are collected into the eph_rest array. The satellite health is one of these variables, which will be needed later on.

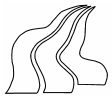
After the new ephemerides are established, the m-file MakeEphFiles is engaged. This updates to files eph.wk1 and eph_rest.wk1, with any new ephemerides. Eventually they will contain ephemerides for all satellites, and will therefore be the equivalent to almanac data.

4.6 TIME CHECK AND CONVERSION

The GPS system, and all GPS-related computation works with seconds of week (sow). As described in the ICD-GPS-200c p. 98 the range must be within ±302400. It is therefore essential that all measurements are tagged with a corrected time in sow.

The given time in the measurement data are units of 50ms, modulo 30 minutes, meaning a time reset every 30 minute. That gives the range 0-36000 in units of 50ms. The conversion factor from units to second is 0.05. Before the time can be used it must be con-

[1] See also Matlab help function for fopen.



verted into sow, and checked for correct range. This is done in the m-file TimeCheck.m, by calling the function AshT2sow.m and Check_T.M.

AshT2sow

Description:

This m-file converts the measurement data time tag in units to GPS seconds of week.

Procedure:

It was first experimentally proved, that the Ashtech time, was synchronised with GPS time. That means that a reset of Ashtech time would fall together with a reset of GPS sow, which happens every Sunday 00:00 GPS time. This knowledge is use in the conversion of the Ashtech time to GPS sow.

The first problem to solve is how to get the full number of Ashtime units since the GPS sow reset. Because every time the Ashtech time encounters a reset, the time is reduced by 30 minutes or 36000 units of 50ms. To compute the full time in units, since GPS sow reset, information about the number of Ashtech time resets NOR must be obtained.

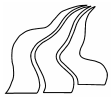
This is done by using the PC-clock. The m-file establishes the hour, minute and sec, at the given time, by running common Matlab functions that reads the pc-clock. The day of week dow, is made by the function dayofweek.

Then the numbers of Ashtech time resets are calculated, at the time given in the measurement data, by counting the numbers of half-hour's passed since Sunday 00:00.

Then the measurement data in Ashtech time, are summed with the NOR * 36000, to obtain the full Ashtech time in units, since GPS sow resets. Afterwards full Ashtech time is converted to GPS sow by multiplication with the conversion factor 0.05.

Before the function returns GPS sow, the time is checked by Kai Borres m-file Check_T, that checks and corrects the range of sow.

If a mistake is made in the time conversion, it will always be wrong by a whole number of half-hour's. The time will therefore either be correct or very wrong. Since no major time errors are observed, the time conversion has proved to be correct. It must be noted, that the above method could encounter problems within one epoch close to the time 00 or 30 minutes. If the measurements originate from one half hour, and the time conversion happens in another, the NOR will be wrong. The PC clock must therefore be set to official GPS time, with a max clock offset 5 sec, when the program is running with an output rate of 5 seconds, to prevent this problem from occurring. Should one or more positions be obscure, they will be discarded by the m-file PosPlot which will return PosOk = 0. (Meaning position not ok!)



4.7 OBSERVATIONS

Before the receiver position can be calculated, some observations must be corrected and assigned. This is done in m-file MakeObs. Here all the observations (C/A code transmit time) are multiplied with the speed of light, to become pseudo ranges, and assigned to the (No. of satellites x 1) array obs. Hereafter obs contains the pseudo ranges from all available satellites to the receiver. The PRN numbers are assigned to a similar array Sats, sorted in the same order as obs. Finally the corrected measurement time in sow, for the epoch, is assigned to the variable Time. Keeping in mind, that the ephemerides are already assigned to the array eph, everything is now ready for the final computation of the receiver position.

4.8 RECEIVER POSITION COMPUTATION

To calculate the receiver position using the least square method, on measurement data from one epoch, quite a few things needs to be computed. The m-file recpo_ls therefore calls a number of other m-files during its execution. All the m-files in this computation are made by Kai Borre. The functions will be described briefly, in the following.

Recpo_ls

Functionality:

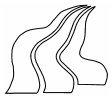
Calculate receiver position from one epoch using least square method.

Procedure:

The method of an iterative least square process needs a preliminary guess, for the receiver position. It is amazingly sufficient enough for the calculation when this is set to zeroes.

The first necessary step is to assign the right ephemeris to each measurement. This is done by the function find_eph, which returns the columns with the corresponding ephemeris from eph, for a data set defined by sats and Time.

The second thing needed, is a time correction. The time available in the data set is the receiver time when data is received. This needs to be converted to the time of the data transmission from the satellite. The correction is therefore a function of the signal travel time. Another correction is also needed at this point. That is the error introduced as a function of the earth's rotation. The correction for earth rotation is made by the m-file e_r_corr. A time check by Check_t.m is also engaged.



When the time is corrected, all satellite positions XYZ in a ECEF[2] system, at time of signal transmission, can be calculated, from the information in the ephemerides. This is done by the m-file satpos

When the satellite positions are known, their azimuth, elevation and distance can be calculated. This is done in the m-file Topocent

With information about elevation it is possible to make a correction for the delay in the troposphere. This is done in the m-file Tropo, which uses a model of a standard troposphere. No meteorological measurements are made; all parameters are therefore either standard values or set to zero. There will be made no correction for the delay in the ionosphere. It is therefore also set to zero.

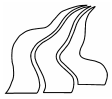
When all iterations are done, the function returns the receiver position POS, the satellites elevation EL, and the Gdop. Nonetheless, more information is available within the functions. For instance the receiver clock offset and the azimuth and distance to the satellites etc. This will be useful information later on, when different calculations like the satellites orbits are made.

It has therefore been decided to modify Kai Borres m-file at one point. The returned values of the function have been expanded so more variables are available outside the function. With this variables available, the next step would be to process them, according to the chosen plots etc. that will be part of the monitor.

4.9 DATA PROCESSING

It have been decided to establish a number of plots, which will visualize the chosen elements according to the analysis. These plots will all be described below, so will the functionality and procedure regarding the specific m-files. Furthermore all plots were tested simultaneously for a period of 1 hour with an epoch interval of 10 sec. The result of this test will also be shown and discussed, for each plot. The plots are described in the same order, as they are computed in the software, because some m-files are dependent on others.

.



4.9.1 Preparing the Plots

The backgrounds for some of the plots need to be established, before the actual plot can be engaged. This is done in SetPlots.m, which is run every time the ephemerides are updated.

Setplots.m

Functionality:

First the satellites above elevation mask plot (salmPlot) is run. Then it clears the figures for the SubSatPlot and StereoPlot and then redraws a new background for the plots. This way the plots are cleared and new backgrounds are drawn, every time a new satellite appears

Procedure:

This m-file's only function is to call three other m-files in the following order.

1. SalmPlot; (Explained in section (4.6.2))
2. DrawlinesSub – Here the sub function SetSubMap is called
3. DrawlinesStereo

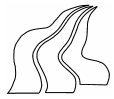
DrawlinesSub

Here the background map for the SubSatPlot is prepared by running SetSubMap.m. In this m-file the array topo is loaded. This is an image of the world in geographical coordinates in γ and λ already defined in Matlab. The image, defined by a 180x360 matrix, is reshaped so the Greenwich is placed at 180 degrees longitude, and equator is set at 90 degrees latitude. This gives Greenwich and equator a central position on the map, and hereby prepares the axes to the following computations. The colours are also redefined into one for water and one for land.

Then predicted orbits lines for the current satellites are plotted on to the map. This is done by using the "almanac" data, retrieved from the ephemeris file, and the array AllSatPos created when making the SalmPlot. It is simply a run similar to the SubSatPlot, with data for every 5 min of the current day, for all tracked satellites. This way a dotted line occurs where the sup satellite point eventually will travel.

Then the next function DrawlinesStereo is run. Here the Stereographic plot is cleared, and new predicted lines are plotted for the current tracked satellites. This is also a run similar to the StereoPlot.m, with input of "almanac" data, and AllSatEl, and AllSatAz created in the SalmPlot.m.

All mentioned plots and arrays are further descried below.



4.9.2 Satellites Above Elevation Mask (SALM)

This plot shows all satellites that has been or will be available within current day of week.

This plot is computed and redraws after every ephemeris update. There are to reasons for this. First the plot will be updated with the newest almanac data, even though this is not really needed. Second and more important three arrays will be made or updated in the procedure, which are needed in the SetPlots m-file which also is run after every ephemeris update. These are AllSatPos, AllSatEl, and AllSatAz.

SalmPlot.m

Functionality:

Makes the Salmplot (Satellites above elevation mask Plot), and establish AllSatPos, AllSatEl, and AllSatAz.

Procedure:

First the day of week dow is computed by the function dayofweek, and start time is computed by multiplying dow with 86400 (24*60*60). Then GPS time in sow is established for the previous midnight.

Then satellite positions, azimuth and elevation are computed for all satellites in the almanac, for 24 hours, with an interval of 5 minutes. The reason for this rather small interval is for the other uses of the array. This is done by running Borres m-fiels, satpos.m and troposent.m. For the first run, a known position is used as input for tropocent.m, herefter the mean position is used. This is computed by running the function meanpos. The positions, elevations and azimuths are all assigned to arrays, for later usages.

Subplot 1 SALM individually displayed

Here the array called Elarray, is computed. It is a made from the AllSatEl, but it only contains zeros, ones and twos. Zero for indicating a space between lines. One for a satellite below the elevation mask, and two for one above.

The array contains 3 similar lines for every satellite and one extra line of zeros between them. When the array then is displayed using the image function, the proportions are just right, and the plot looks nice.

Suplot 2 SALM displayed merged.

Here the MElarray is computed from data in AllSatEl.

This time all values above the elevation mask in AllSatEl are gathered in the beginning of the row. When the array is then later displayed by image, it will show a merged im-

age of the pervious subplot. This way it is easy to see how many satellites can be expected above the elevation mask for the current day.

Result

The result of the predicted satellites for 30-5-02 is shown below.

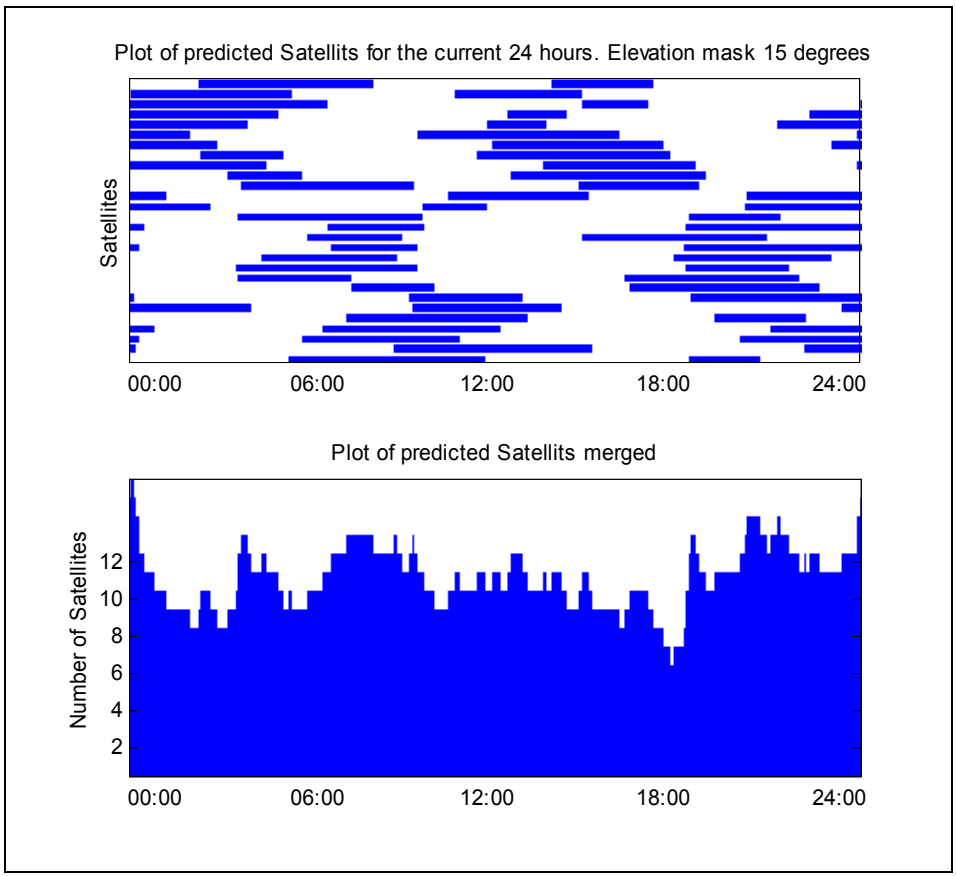
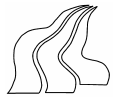


Figure 15 Plot of Predicted Satellite above elevation mask Plot. (Aalborg 31-05-02 11:40 to 13:00)

Comments

The plot shows that the predicted satellite coverage for this day should be outstanding. Only for a short period around 18:00 did the plot predict les than 8 satellites. For a few moments there are actually more than 12 satellites available on the sky. This is too little use in practice though, since most geodetic receivers like the ZX have 12 channels. Further more chances for one or more satellites being blocked by objects are quite great. For the given measurement the number of tracked satellites varied from 5 to 7, which is the minimum for good computations.



4.9.3 Important Values

The purpose with this plot is to visualise the values important for the accuracy. In the analysis the group found that these values would be the DOP values, satellites health and signal to noise ratio S2NR.

ValuesPlot.m

Functionality:

Plots the DOP values, satellites health and signal to noise ratio S2NR.

Procedure:

The DOP values are already computed in the least square computation, and the health and S2NR are available in the measurements. The plot just visualise this values by using the image function to draw bars according to the value. The DOP values are drawn in the range 0 – 6 with a blue bar. The health for satellites on the 12 channels, are drawn as a green square for good health, and red for bad. The S2NR for satellites on the 12 channels, are drawn as blue bars in the range 0 – 60.

The values plot is updated for all epochs, and a value with the number of satellites with good health is computed.

Result

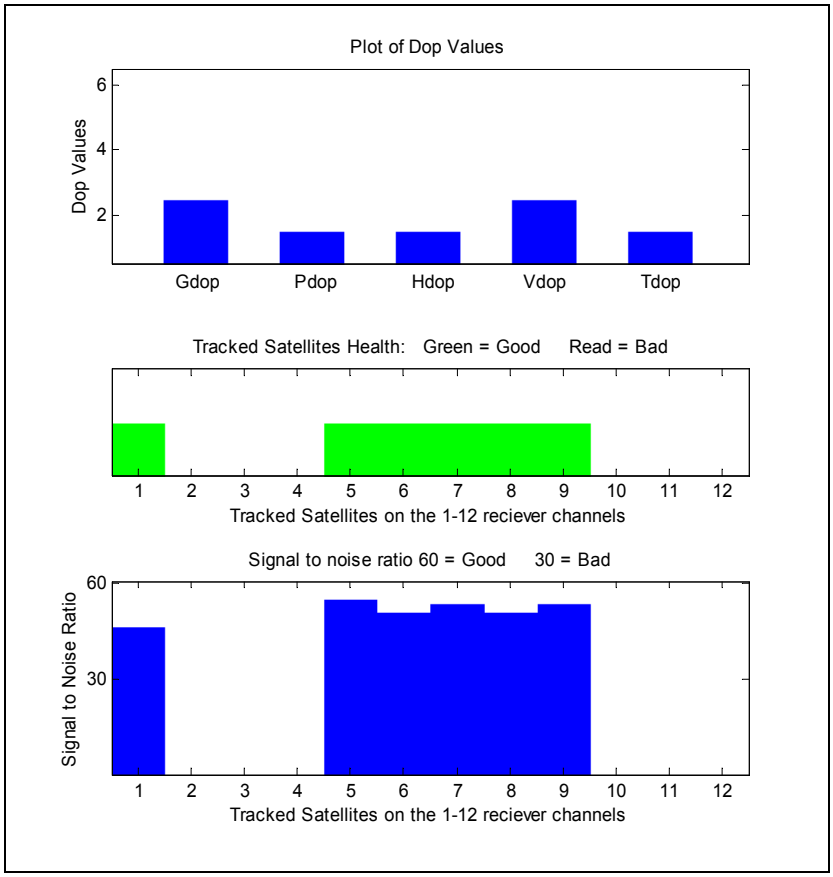


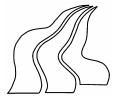
Figure 16 Exampel of Values Plot. (AAborg 31-05-02 01:00 Pm)

Comments

The plot shows good DOP values as expected, and did at no point during the run get close to the maximum of five. Neither did the signal to noise ratio go below 45 db/Hz which is the limit. No satellites came up with a warning concerning its health during the test period.

4.9.4 Receiver Position

The group has decided to plot the receiver position in reference to a known position, calculated as the mean of all previous positions. The actual plot will therefore be the difference between the current mean and the computed receiver position. There are several reasons for this choice. First it is not intended with this monitor, to display just one accurate position, but rather to visualise the running accuracy of the system. This is done by displaying a difference instead of an actual position. This way the specific plot automatically shows the deviation of the position accuracy, and the plot will be distributed around 0.0.



The plot is defined within a 6x6 m square, to ensure a proper zoom factor. To keep the plot within this range, the mean position and the plot will be reset, if the position floats away from the mean by more than ±3 m in X or Y, for a specific number of times. This also cleans up the plot, which otherwise would be full of points without any reset.

The current plot is accompanied with an error ellipsoid, created by the variance on x and y and there covariance. The ellipse is scaled equally as the axis. It is therefore directly comparable with these. To further highlight the current point, the small and large axes of the ellipsoid are also displayed. The m-file that creates this plot is called PosPlot.

PosPlot.m

Functionality:

Plots the receiver position in reference to the best mean position.

Procedure:

The first purpose of this m-file is to compute the mean receiver position X_hat_k and Y_hat_k, for the epoch k.

This is done according to the theory for recursive update of a static one-parameter problem, where the following formula is described. (Strang & Borre 97 p. 546)

$$\hat{X}_k = \frac{k-1}{k}\left(\frac{b1+\cdots+b_{k-}1}{k-1}\right)+\frac{1}{k}b_k \qquad \textbf{(4.1)}$$

Where:

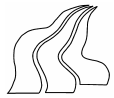
$b1+\cdots+b_{k-}1 = X\_sum$ (X_sum is the sum of all previous values)

k = current epoch and

b_k is the current measurement.

Afterwards the reference position is calculated, by subtracting the current receiver position from the updated mean position.

If the difference is within the range of ± 2m in x, y, it is plotted on a 6x6m plot. It has to be a rather small plot, to give the right zoom. If it is not within this range, the computations of the epoch are discarded, and a warring message is displayed. Furthermore, a variable PosOk, is assigned from 1 to 0. This variable is used as an accuracy verifier, before updating any of the plots, except the values plot. If an epoch is discarded more than 5 times, the mean receiver position is reset, and the plot is cleared. This is done to make sure, that the plot is centred on the best mean, and containing the new plotted values.



The ellipsoid is drawn by a modified m-file ellconf.m made by Kai Borre. It has been modified, so it plots the ellipsoid on the position plot with the current point as centre. The input to the file is a 2x2 array from the least square computation, containing the variance for X and Y, and there covariance. The plot is only updated, if the accuracy value PosOk is 1.

Results

When the plot was tested there were no reset functionality, and the axes were set to auto. The reason for this was to analyse the behaviour of the plot over time. Here is the plot showing the result of the test.

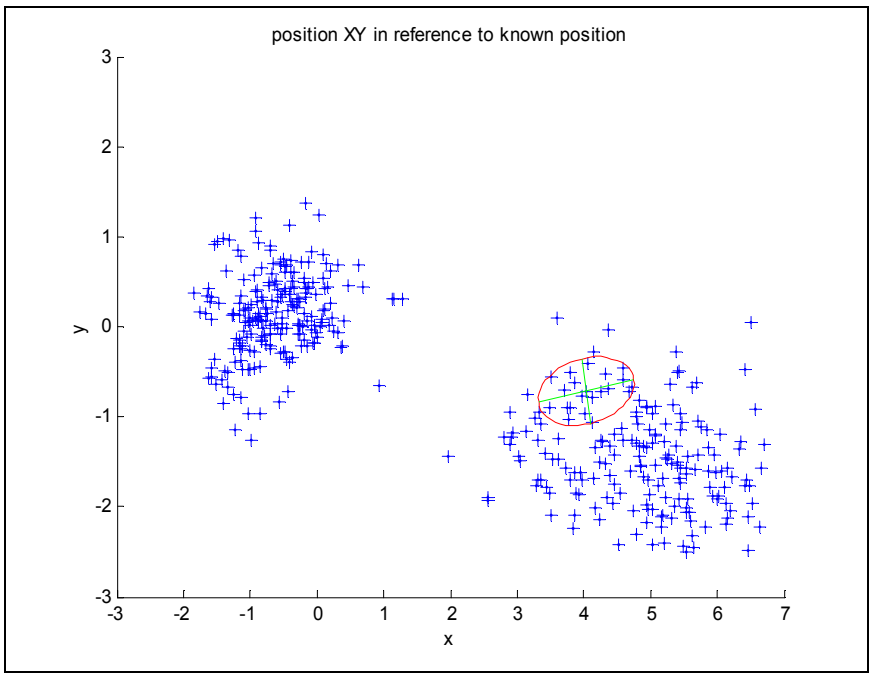


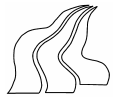
Figure 17 Test of Position Plot. (AAborg 31-05-02 11:40 to 13:00)

Comments

The position was concentrated in the range ±1½ for x and y in the beginning of the test. At a certain point the position of the plot shifted to the range 3 - 6 for x and 0 - -2 for y. The behaviour was repeated throughout later runs. It clearly shows that the receiver's position for a certain constellation of satellites is concentrated around a mean position with a spread of about ± 3 meters in X and Y. Another satellites constellation gives another mean with the same spread, but randomly shifted a few meters. This is the reason for giving the plot the size of 6x6 meters, and introducing a mean position reset.

4.9.5 Stereographic Plot of Satellites Orbit

To establish a stereographic plot of the available satellites, seen from a receiver position point of view, the Matlab function polarhg is used. This function is a modified version of the conventional function polar, used to plot polar coordinates defined by azimuth and radius.



The plot is not used to plot conventional polar coordinates, but shows the satellites as a function of their elevation and azimuth. The scale on the x and y axis is therefore defined as the elevation between 0 to 90 degrees, with 0 equal to a satellite in zenith. The azimuth is defined as the angel from north to the satellite in the range 0 – 360 degrees.

This skyplot gives a panoramic image of how the available satellites, are distributed around and above the horizon.

StereoPlot.m

Functionality:

Plots the satellites position as a function of elevation and azimuth.

Procedure:

The elevation and azimuth is computed with the m-file tropocent, with the receiver mean position meanpos, and SatsPos as input. A check is made to ensure, that the mean position is based on at least 3 computations.

The elevation, is traditionally defined in the range (0:90), with 0 as the horizon. Before the elevation can be used as input to polarhg.m, the range is inverted by subtracting the angel from 90 degrees.

The satellites are marked with a colour identified by the satellites unique PRN number. This procedure is repeated throughout all plots, making the marker colour for one satellite the same, in all given plots. Every time the plot is updated, the previous points are deleted, before new are added. The plot is only updated, if accuracy value is ok.

Results

To show all the points for the test period, the previous plotted points were not deleted during the test.

Figure (18) shows a sample of the stereographic plot with seven tracked satellites. The thin dotted lines are the predicted skyplot of the tracked satellites, made by DrawlinesStereo.m.

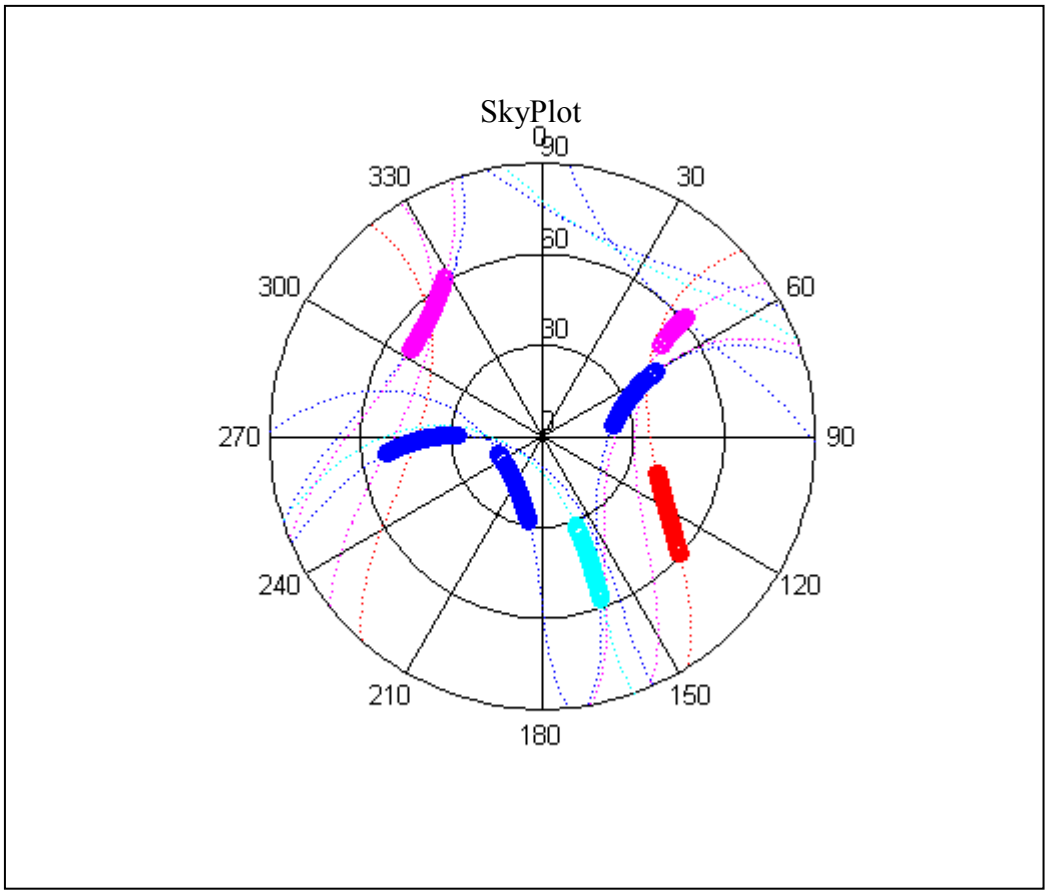


Figure 18 Example of Stereographical Plot. (Aalborg 31-05-02 11:40 to 13:00)

Comments

It can be seen on the plot that the path of the current tracked satellites actually follow their predicted path. If the plot is run for a longer period of time, the plot would show the cut-of-angel on 15 degrees. (No tracks would get beyond a radius of 85.)

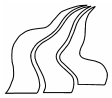
4.9.6 Sub Satellite Point

To visualizes the satellites position on a world map, it is necessary to project the positions vertically down to the surfaced of the earth, called the sub satellite point **SSP**, and then into a chosen map projection.

It is not necessary to calculate the position of sub satellite point very accurate, because it is only needed for a simple visualization, with a rather big dot on a low resolution world map. A simplified solution is therefore used in the computation of the sub satellite point

For the world mab, a Mercator projection was chosen. Primary because this projection, with positions in longitude and latitude, is worldwide used and understood.

SubSatPlot.m



Functionality:

Plots the Sub satellite position **SSP** on world map

Procedure:

The satellites cartesian coordinates XYZ are part of the output from Recpo_ls.m in the array SatPos. These positions are converted into sub satellite points by making a unit vector, derived from the vector defined by the XYZ, and then this unit vector is scaled by the radius of the earth. This gives a simplified solution for the Sub satellite point in cartesian coordinates.

The error introduced by this method, comes from assuming that the earth is a sphere with a constant radius, when it is really an ellipsoid. Because it is a vertical projection down on a sphere instead of an ellipsoid, this method especially gives an error on the height. Since only the horizontal coordinates are needed in the map projection, this error is disregarded. Furthermore the errors introduced, are zero at equator, and max by the poles. Because of the nature of the satellite orbits, the sub satellites point never gets close to the poles, because it never gets above the ±55 degrees latitude. This means that the error introduced in the plain coordinates generally are quite small, compared to the rather big dot, used in the plot.

This method is therefore acceptable, its simple purpose taken into account.

After the sub satellite point has been computed, it is converted, by the m-file Ashc2gm, into geographical co-ordinate's defined by phi in the range [-90:90] and lambda in the range [-180:180]. Then the SSP is plotted onto the map. At the same time, the previous plotted SSP are deleted. The plot is only updated, if accuracy values are ok!

Result

A plot of the Sub satellite point is displayed below. The individual points have not been deleted, so all points for the test period are visible.

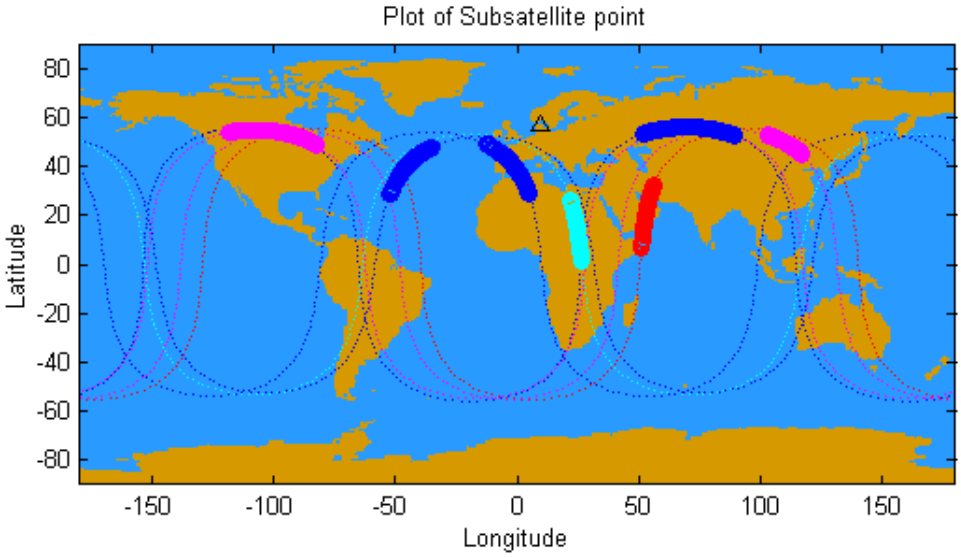
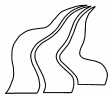


Figure 19 Exampel of Subsatellite point plot. (AAborg 31-05-02 11:40 13:00)

Comments

Here the plotted sub satellite points all follow the predicted lines. If the plot were run for a long period of time it would show all the points where the satellites rise and go below the horizon. If you compare the plot with the stereographical plot, the same satellites can be identified by their colour, position and track.



4.10 DATA HANDLING OFF-LINE

All files that is specifically designed to work on-line, has replicas called XXXoff.m, which will work offline. Every time the monitor runs online a time file timefile.dat and a datafile MBNVar.dat are created if wanted. These files will simulate the online data stream collected offline. The group made these files primarily for debugging and developing purposes. For simulating a specific amount of data, the Matlab work directive must be set to the data location and the m-file gpsoff must be executed.

The off-line files are lined up in the table below, but will not be described further, since they are not part of the primarily goal of this project. Anyway, they are not much different from their origins, since only details concerning the com port, files and time are changed.

Off line m-files	**Description**
Gpsoff	Runs the main program offline
ReadMBNVarOff	Gets data from a file
CheckTimeOff	Checks the time offline, by using a time file.
AshT2sowOff	Makes the conversion from units to sow offline
SetAllVarOff	Sets all needed parameters and variables offline

Figure 20 List with the special off-line files

5 SYSTEM TEST

After the system had been fully debug, and seemed to run without problems, a full system test was engaged. This was to show if all components would work well together and according to the specification. The ephemeris files, already in the system, were deleted to simulate a cold start. The system was set to run with an update rate of 30 sec after the initialization. This was to reduce the number of computations, and the file size of the measurement data. Further more for a re-run of the stored data off-line, it would only be practical with a reasonable number of epochs.

It turn out that the system worked well without problems, and the ephemerides for all 28 available satellites were obtained during the next 20 hours. All plots worked well according to the specification. This can be seen in the following figure.

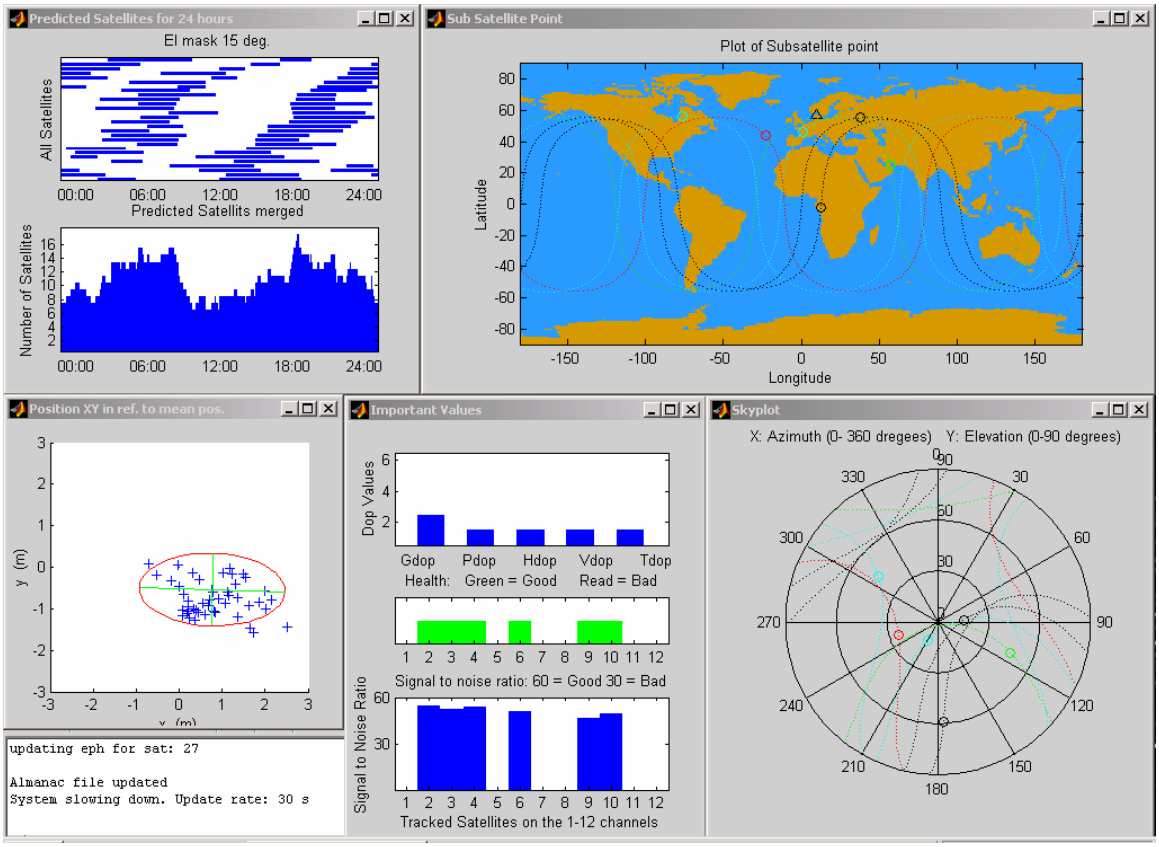
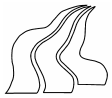


Figure 21 Screen domp during the test.

The plot of the receiver position in reference to the best mean position (PosPlot) was more or les reset every time new satellites were being tracked, or some disappeared. The system also encountered several corrupted lines of measurement data, and there was a period with less than 5 healthy tracked satellites. Correct warnings were displayed accordingly.



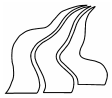
The system proved to be quite stable, and because of the many try, catch and end loops the closedown procedure requires some patience.

A full Matlab profile report of the system test with a file list can be found in the appendix (B), together with a plot of the time consumed by the most used m-files see appendix (C). This plot has been produced both for the full test, and also for a short run with only 3 epochs after initializing, to show a more true picture of the m-files profile. The one for the full test is highly inflicted by the long wait for new data between each epoch.

Further more a log file was made during the test run. It is rather long (16p), so only the content will be given here.

Massages during 20 hour run	**Number**
Almanac file updated	41
Position plot resets	43
Corrupted Measurement lines	26
Epochs with less than 5 healthy satellites	34 (All at one time)

Figure 22 Massages from the log file



6 CONCLUSION

The aim of this project was to address the issues stated in the problem formulation section (1.4). It can be stated that we were able to:

- Establish serial communication with the receiver
- Explore and handle the received data according to specifications
- Complete the needed computations, required by the chosen monitor features.
- Explore visualisations opportunities and chose from these a way to present the computations graphically in real time.

This clearly shows that the group have been able to solve the apparent problems stated in the problem formulation.

With regard to the more theoretically orientated questions, we can also conclude that the group during the project found a suitable way of making a GPS-monitor and interface to the ZX, with the purpose of analysing the coordinate accuracy.

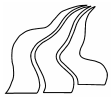
A more reasonable way to represent the graphics would have been to incorporate the plots in at GUI. This would also have allowed us to show specific numbers like standard deviation, GPS week, second of week and UTM time. These features were not implemented because time did not allow the group to experiment with a GUI.

Furthermore the group found what would be the most interesting features to include in the monitor/ interface, to obtain knowledge about the factors that influence the coordinate accuracy.

However, additional features could also have been added. The monitor is not cable of showing what happens over time, because only current values are shown. A feature where one could analyse the statistical values for a full period of time would have been preferable to include.

Regarding the program even thought the final product is working without problems, some things should be improved. Many of the assigned variables and arrays are presumably the same, or could be designed so they could do more than one function. A lot of the m-scripts could also be designed as functions, which would have made the software design more consistent. All in all the system could use a thorough check up, according to the mentioned problems.

For serious GPS monitoring, the software would have to be improved to take care of all the issue addresses above. Future works could concentrate on these issues.



7 BIBLIOGRAPHY

(Strang & Borre 97) Gilbert Strang and Kai Borre. Linear Algebra,deodesy, and GPS , 1997. ISBN: 0-9614088-6-3

(Gunter Seeber 1993) Satellite Geodesy. Foundation, Methods and Application.

(Teunissen $ Kleusberg 1998) GPS for Geodesy

(Tiberius 2000) Christian Tiberius. Mathematical Geodesy and Positioning. Delft University of Technology. Lecture Notes. April 2000.

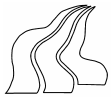
(Mathworks) Mathworks. The Serial Port Interface, http://www.mathworks.com/access/ helpdesk/help/toolbox/instrument/ ch_vis10.shtml, 2002. URL visited: 02/2002.

(GPSWORLD 2002) GPS World. World Wide Web, http://www.gpsworld.com/gpsworld/, 2002. URL visited: 03/2002.

(The GPS 2002) The Global Position Systems. http://www.colorado.edu/geography /gcraft/notes/gps/gps_f.html. 3/27/00. URL visited: 03/2002.

(ZX. Ref.Manual 2002) Z-Xtreme GPS Receiver. Operation and Reference Manual, 2002.

(Part-Enander etc) The Matlab Handbook



8 WORKING PROCESS

The following section describes the working process, including our approach, method and procedure used in this project to achieve our goal.

First and foremost, it would be worth mentioning the rough start of the semester. The first 3 weeks of the semester was wasted because the group was not yet formed. On the fourth week we were able to meet as a group and the project was for the first time introduce to us. We were at this time also allocated a group room. The project then started.

Brainstorming led to literature review. This made us gain insight into the problem. After getting an over of the problem we then proceeded to define our task and to prioritize them. The first task was to establish a communication link with the receiver through Matlab serial interface. This took us a longer time than schedule since the group members were all novices in programming with Matlab. In fact we had serious problems unpacking the binary data bitwise in Matlab. However, after struggling with this problem for a long time, help finally came from our supervisor.

The project then proceeded with writing of Matlab scripts that could handle and process the data. The second problem was the use of complicated m-files already constructed by Professor Kai Borre. Though the files were very helpful, we needed to spend some time trying to understand all files and there functions before we could use them.

After being able to establish communication with the receiver we then proceeded into developing the software for data handling and processing. Documentation was started at this time. Testing of Matlab scripts were done each time they were written and incorporated in the whole program. The testing were herby done as part of the development.

After having struggled with binary data and Matlab code, things cleared up as we went along, and in the end of they day, we were able to finish a piece of software, how crude and simple it might seem, that could for full our own expectations.

All in all, the problems introduced in this semester, have been most interesting and have court our full attention.

END

9 GLOSSARY

Almanac

A data file that contains orbit information on all satellites, clock corrections, and atmospheric delay parameters. It is transmitted by a GPS satellite to a GPS receiver, where it facilitates rapid satellite vehicle acquisition within GPS receivers. Almanac data must be acquired before GPS navigation can begin.

Baud rate

Baud rate is the number of bits transferred per second

C/A Code

The standard (Clear/Acquisition) GPS code, also known as the "civilian code" or S-code. The Code is a spread spectrum direct sequence code that is used primarily by commercial GPS receivers to determine the range to the transmitting GPS satellite.

Delusion Of Precision

Delusion Of Precision An indicator of satellite geometry for a unique constellation of satellites used to determine a position. Positions tagged with a higher DOP value generally constitute poorer measurement results than those tagged with lower DOP.

ECEF

Earth centred earth fixt coordinate system.

Ephemeris (Ephemerides in plural)

The predictions of current satellite position that are transmitted to the user in the data message.

Global Positioning System (GPS)

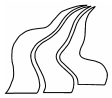
a system for providing precise location which is based on data transmitted from a constellation of 24 satellites

Latitude

A north/south measurement of position perpendicular to the earth's polar axis

Longitude

An east/west measurement of position in relation to the Prime Meridian, an imaginary circle that passes through the north and south poles.



Mask Angle

The minimum acceptable satellite elevation above the horizon to avoid blockage of line-of-sight.

Mercator projection

A Mercator projection is a mathematical method of showing a map of the globe on a flat surface. This projection was developed in 1568 by Gerhardus Mercator a Flemish geographer, mathematician, and cartographer. The projection saves the shapes and directions, but introduces a size distortion.

Pseudo-random noise (PRN)

A signal with random noise-like properties. It is very complicated but repeated pattern of 1's and 0's.

Pseudo-range

A distance measurement based on the correlation of a satellite transmitted code and the local receiver's reference code, that has not been corrected for errors in synchronization between the transmitter's clock and the receiver's clock.

Selective Availability

Selective Availability (S/A) Intentional degradation of the performance capabilities of the NAVSTAR satellite system for civilian use by the U.S. military, accomplished by artificially creating a significant clock error in the satellites. (web www.navtechgps.com)

(http://www.navtechgps.com/glossary.asp)