UNIVERSITY OF CAPE TOWN

Using GPS bistatic signal for land and ocean remote sensing in South Africa

by

Shikoane Given Phaladi

A DISSERTATION
SUBMITTED TO THE FACULITY OF SCIENCE
IN PARTIAL FULLFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS AND APPLIED MATHEMATICS

CAPE TOWN, SOUTH AFRICA

JUNE, 2007

Using GPS bistatic signal for ocean and land remote sensing in South Africa

Shikoane Given Phaladi

June 1, 2007

Abstract

Traditionally, direct GPS signals are used for navigation and positioning, while the indirect reflected signals are considered a nuisance. However, recent studies show that indirect reflected signals contain some useful scientific data. GPS reflected signals introduce a new and exciting way of doing ocean and land remote sensing, and have even more advantages over traditional remote sensing tools. This project discusses the basic principles and theory of this new technology, and concentrates on reflection points and Fresnel zones. The GPS receivers are placed at different coastal regions within South Africa, and the simulation of the reflection points and Fresnel zones are observed as the GPS satellites pass over South Africa. The East London area was chosen as the location to place the receiver throughout my analysis. Areas of the Fresnel zones reaching a maximum of about 6500 km^2 were observed at different receiver heights and the software to make these simulations was written using the IDL language. Results shows that this new tool of remote sensing is feasible and has potential to be used in South Africa. The uses for this new tool include ocean altimetry, ocean, land, ice sheet remote sensing etc.

Acknowledgments

It is with a great deal of pleasure that I acknowledge my heartfelt thanks and gratitude to my supervisor, Prof. Mike Inggs for his support, encouragement and guidance throughout my research. I would also like to thank Dr. Richard Lord for allowing me to use and edit some of his IDL codes for my project, and for his helpful comments and discussion. I also wish to thank my fellow graduate students, staff members of National Astrophysics and Space Science Program (NASSP) and Radar Remote Sensing Group (RRSG) for providing a pleasurable and conducive environment for studies and research.

Regarding my friends and family, I wish to thank my mom and dad for their consistent emotional support, encouragements, motivation and driving force throughout my studies and this project. Above all I would like to thank God Almighty for the life and divine health He blessed me with. My project has been funded by National Research Foundation (NRF) and NASSP. For that I am also most grateful.

Contents

1 Introduction 9

2 Global Positioning System (GPS) 13
- 2.1 Advantages of using GPS 15
- 2.2 The GPS ranging signal 16
- 2.3 The GPS receivers and antennas 18
  - 2.3.1 The Delay Mapping Receiver (DMR) 19
  - 2.3.2 Digital Beam-Steering GPS receiver 20
- 2.4 Summary 21

3 Geometrical Theory 23
- 3.1 GPS bistatic geometry 23
  - 3.1.1 Bistatic GPS signals 25
- 3.2 Smooth or Rough-Surface Criteria 26
- 3.3 Multipath Theory 28
- 3.4 Fresnel Theory 28
- 3.5 Effect by Earth's curvature 31
- 3.6 Summary 31

4 Global Coordinate Systems and Algorithms 33
- 4.1 Terrestrial and Inertial Reference Systems 34
  - 4.1.1 Conventional Terrestrial Reference System (CTRS) . . 34
  - 4.1.2 Conventional Inertial Reference System (CIRS) 35

4.2 Geodetic Coordinates, Geoid, and Datums 35
4.2.1 Ellipsoidal Coordinates 36
4.2.2 World Geodetic System 1984 (WGS 84) 38
4.3 GPS Orbits and Satellites Position and Velocity 39
4.3.1 GPS Orbit parameters 39
4.3.2 Satellite Position and Velocity 40
4.4 Other Considerations 42
4.5 Determining the locations and the size of the active scattering region 43
4.6 Summary 44

5 Results and Discussion 45

6 Conclusions and Recommendations for Future Work 51

List of Figures

2.1 EM spectrum 16

3.1 Geometry of the ocean-reflected signal 25
3.2 Correlation Power vs Code Chips 25
3.3 EM reflection from surface 27
3.4 Fresnel zone on a reflecting plane 29
3.5 3D Geometry of GPS Surface Reflection 29

4.1 Terrestrial and inertial reference systems 34
4.2 Cartesian and ellipsoidal coordinates 37
4.3 The Fresnel zones on a reflecting plane 43

5.1 Cylindrical map showing the GPS satellites coverage 45
5.2 Orthogonal projection map showing the GPS satellites coverage 46
5.3 Cylindrical projection of the Southern African (RSA) map. . . 46
5.4 Durban GPS receiver placed at 1km height 47
5.5 East London GPS receiver placed at 1km height 47
5.23 Area of the Fresnel zones for receiver at 100km 47
5.6 George GPS receiver placed at 1km height 47
5.7 Table Mountain GPS receiver placed at 1.1km height 48
5.8 GPS satellite latitude and longitude tracks 48
5.9 Semi-minor of the Fresnel zones for receiver at 1km 48
5.10 Semi-major of the Fresnel zones for receiver at 1km 48

5.11 Area of the Fresnel zones for receiver at 1km 48
5.12 Semi-minor of the Fresnel zones for receiver at 3km 48
5.13 Semi-major of the Fresnel zones for receiver at 3km 48
5.14 Area of the Fresnel zones for receiver at 3km 48
5.15 Semi-minor of the Fresnel zones for receiver at 10km 48
5.16 Semi-major of the Fresnel zones for receiver at 10km 48
5.17 Area of the Fresnel zones for receiver at 10km 48
5.18 Semi-minor of the Fresnel zones for receiver at 20km 49
5.19 Semi-major of the Fresnel zones for receiver at 20km 49
5.20 Area of the Fresnel zones for receiver at 20km 49
5.21 Semi-minor of the Fresnel zones for receiver at 100km 49
5.24 Semi-minor of the Fresnel zones for receiver at 6500km 49
5.25 Semi-major of the Fresnel zones for receiver at 6500km 49
5.26 Area of the Fresnel zones for receiver at 6500km 50

Chapter 1

Introduction

Remote sensing is the science of acquiring information (sensing) about the object (i.e. Earth's surface) without being in contact with it (remotely). This is done by sensing and reading reflected or emitted energy and processing, analyzing and applying information. Without direct contact, some substitude method must be utilized for gathering and transferring this information. Since the oceans are a particularly hostile environment, with frigid temperatures and potentially bone-crushing ambient pressures, remote sensing applications are particularly attractive to researchers. Remote sensing devices enable one to safely obtain vast amount of geophysical data without getting one's feet wet or digging the ground.

This project explores a new and exciting way of doing remote sensing using Global Positioning System (GPS) as a tool. Each satellite broadcast signals in different carrier frequencies [French, 1996], which are the popular L1 (1.57542 GHz) and L2 (1.2276 GHz),and not so popular L3 and L4 (to be discussed in section 2.2). This range of frequency is optimal for soil moisture remote sensing [Masters et al., 2000]. Recent studies also show that reflected signals over the ocean surface carry information about status of the ocean (i.e. wave height, surface roughness, wind speed, wind direction

etc.)[Komjathy et al., 1998]. The satellite and the receiver make the system a bistatic radar system. Chapter two will review recent studies based on the GPS as a new tool for remote sensing. The circular polarization and special structure of GPS signals that provide this unique opportunity will also be discussed under this chapter, together with the types of GPS receivers used to retrieve the signal information.

The resolution of the system is always a big concern for many users in satellite remote sensing technologies. The same gapplies to the GPS system. The signal reflected from the ocean surface originates from the glistening zone surrounding a nominal specular reflection point. The roughness of the ocean contributes to the size and the shape of the glistening zone. The crux of the project is to investigate the regions of the glistening zone that are important in contributing to the total field at a given receiving point. These regions are called the Fresnel zones. Moreover to find the coverage of the surveillance as a function of time, how these vary with satellite and receiver positions, and the measure of signal strength. The above mentioned will be discussed in chapter three, however I will start with the introduction to GPS bistatic geometry. Three dimension (3-D) reflection geometry will be explored in detail within this chapter as well.

Chapter four will summarize the equations needed to get the 3-dimension model of the Fresnel zones, and consider all the relevant assumptions. Based on the equations, the algorithm will be discussed and explored in order to write software; using the IDL language. The simulations will be made using software and the results will be quoted and discussed in chapter five.

Finally chapter six will conclude based on the results that indicate that GPS can be efficiently be used as a new tool for ocean and land remote sensing and its feasibility in South Africa. Future recommendations will be made

to better the system or the current models.

Chapter 2

The Global Positioning System (GPS) as a remote sensing tool

The Global Position System (GPS) represents the fruition of several technologies, which matured and came together in the second half of the 20th century. It is a space-based navigation and positioning system, developed and operated by US millitary [French, 1996]. The GPS consists of a constellation of 28 satellites (24 operational and 4 on stand-by), each satellite transmitting a constant signal down to earth. However the GPS is not the only system of its kind. There is also GLObal NAvigation Satellite System (GLONASS) developed by the Soviet Union (Russia). During its development, the system had a full constellation of 24 prototype satellites broadcasting in 1996, but has since declined [Misra, 2004].

General uncertainty about the future of the system appears to have limited the demand and discouraged manufacturing of the GLONASS related technologies (i.e. receivers). Both the GPS and GLONASS form part of the new system called the Global Navigation Satellite System (GNSS). In addition to GPS and GLONASS systems, the European Satellite Navigation System (Galileo) will form part of the GNSS system as well. Another

system,Galileo could be operational in 2008 [Galileo[0]]. The versatility and availability of signals from the GPS has given birth to many new GPS applications.

Measurements on the direct GPS signals have been succesfully used in navigation and positioning, while indirect and reflected signals were viewed as a nuisance. Ionospheric and tropospheric delays of GPS signals caused by variation of sun's activity (i.e. solar flares) are known commonly to be error source affecting the use of the GPS for positioning and navigation. However space physicists have realised the phenomenon as useful for atmospheric remote sensing. The theory and observation have been explained by [Liu et al., 2004] and [Phaladi et.al., 2005] in their papers about the effects of solar flares on the ionosphere. Another error affecting the GPS signal is the surface multipath, however it has only recently been recorgnized that multipath from the GPS signals reflecting off the sea surface could be utilized as a new tool in oceanographic remote sensing ([Komjathy et al., 2000], [Armatys et al., 2000], [Zuffada, 2002] and [Garrison et al., 2000]).

The strength of the reflected signal is also a discriminator between wet and ground areas, and therefore could be applied to coastal and wetland mapping [Masters et al., 2000]. In both cases, a phenomenon that is usually regarded as an error source to navigation was recognised to contain useful scientific data. Not only can the GPS signal be used for ocean remote sensing, it can also be used for radar target detection and (reflector) change detection [Li et al., 2002]. The use of real-time kinematics GPS (RTK-GPS) to measure the elevation of terrain over an open unobstracted area, are demostrated in paper [Chang, 2004]. It is used as the solution for many time-critical applications such as engineering surveying and high precision navigation and guidance [Chang, a reference thereof, 2004].

[0] http://www.esa.int/navigation/pages/indexGNS.htm

2.1 Advantages of using GPS signal

The GPS remote sensing tool has some advantages over the traditionally used remote sensing tools like radar altimeters and scatterometers. However the system is not a replacement for the altimeter ([Masters et al., 2001] and [Lowe et al., 2002]), but rather advances its usage. Because GPS satellites transmit the signal source, a completely aircraft altimeter based upon GPS bistatic (to be discussed later) would be beneficial for covert operations and applications limiting power and weight [Masters et al., 2001]. The inability to measure mesoscale process, is a most prominent limitation of current radar altimeters [Lowe et al., 2002]. The GPS altimetry would involve an orbiting receiver obtaining time and position information from the GPS constellation, measuring ocean height from the surface reflected signal. "The advantage over monostatic radar altimeters is that the receiver could produce about ten simultaneously measurements (or 20 when the Galileo is operational)" [Lowe et al., 2002], distributed over an area thousands of kilometres across-track [Galileo], thus providing a coverage that is an order of magnitude denser than nadir-viewing altimeters.

The instrument is also a passive device making it relatively inexpensive. Because of the multi-static nature of the GPS observations, our current capability of global sea surface measurements will improve in two important ways: improved spatio-temporal resolution and coverage. (see [Zuffada, 2002] for more details).

GPS signal also offers advantages above the combination of active microwave remote sensing using Synthetic Aperture Radar (SAR), passive microwave sensing, and imaging in optical and thermal wavelength. First, opti-

cal and thermal sensors are limited by cloud cover and visibility conditions. Secondly, for as much as SAR images have sufficient spatial resolution, however "repeat times of existing satellites are relatively long compared to the rate of change of open water fraction in the ice pack" [Komjathy et al., 2000]. Finally, even though spaceborne passive microwave sensors offer more frequent coverage at several wavelengths, they suffer substantially lower spatial resolution.

In comparison to conventional scatterometers, the use of the forward scatter of the GPS signal has several advantages. First, in the absence of a transmitter, (obviously) cost, complexity, size and power requirements will be reduced significantly by an order of magnitude. Secondly, space-qualified and readily available hardware, space flights GPS receivers are presently an "off the shelf" item due to growth in GPS for orbital navigation and attitude determination. Finally, bistatic (forwardscatter) geometry will offer much higher signal strength, which decreases with an increase in surface roughness, incontrast to backscatter which has the opposite dependence. "This dependence way allow the measurements to be made at lower speed conditions than is possible with backscatter" [Garrison et al., 2000].

The primary disadvantage is that in comparison with radar signals, GPS signals are weaker, necessitating larger antennas and/or longer observations.

2.2 The GPS ranging signal

Looking at Figure 2.1, one can notice that microwave spectrum includes different wavelength bands, however for the purpose of the project I will focus on L-band (between 1-2 GHz) since each GPS satellite transmits two microwave radio frequency signals at L1 and L2. These two carriers are coherent and

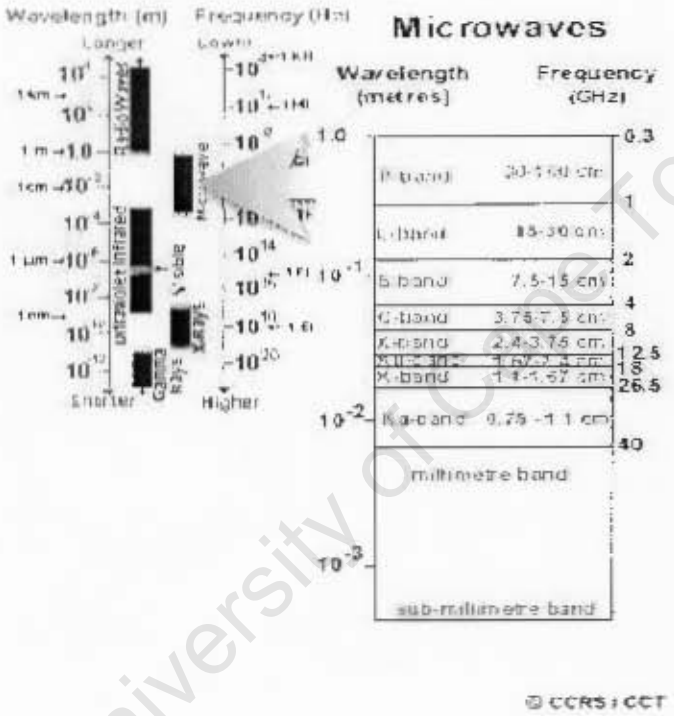

Figure 2.1: The electromagnetic spectrum and its characteristic wavelength bands copied from http://ccrs.nrcan.gc.ca/resource/tutor/fundan/chapter3/02_e.php

modulated by various signals. In this section the structure of the signals sent out by the GPS satellites is examined.

Apparently satellites do not transmit the signals on L1 and L2 only, they also transmit additional radio frequency (RF) signals at frequencies refered to as L3 (associated with Nuclear Detonation Detection System) and L4 (reserved for military purpose) [Misra et al., 2004]. Superimposed on these radio carrier wave signals are Pseudo-Random Noise (PRN) codees that are unique to each individual satellite. The carrier signal is modulated bi-phase, and the modulation programmed in the signal carries important information, sort of like a Morse-code (binary signal).

There are two different pseudo-random code strings used by GPS. They are the Coarse/Acquisition code (C/A code) available to civil users and Precise or Protected code (P-code). The C/A code is used for Standard Positioning Service (SPS) and is only available on L1 carrier, however some civil users requested C/A modulation on L2 carrier to allow for ionospheric calibration [Gibbons, 1999]. Since the two signals are generated synchronously, the user who receives both signals can directly calibrate the ionosphere group delay and apply appropriate corrections. The C/A code is a sequence of 1023 bi-phase modulations carrier wave. Each opportunity for a phase-reversal modulation or switch from a zero to a one, is refered to as a "chip" [Wells et al., 1986]. This entire sequence of 1023 is repeated each millisecond resulting in chip rate of 1.023 MHz [or megachips/s (Mcps)] and wavelength (chip width) of 300 m. Each satellite carries its own unique code string.

On the other hand the military P-code consists of another sequence of plus ones and minus ones, emitted at the frequency 10.23 MHz (10 times that for a C/A code) and wavelength of 30 m, which repeates itself only after 267 days. "The 267 days are chopped into 38, seven-day segments. Of these,

one week segment is not used, five are reserved use with ground stations, called pseudolites, leaving 32, seven-day segments, each assigned to a different satellites" [Wells et al., 1987]. Due to higher wavelength the P-code is more precise and reduces the noise in the received signal. It is used for Precise Positioning Service (PPS). The P-code becomes Y-code when encrypted ([Wells et al., 1987] and [Gibbons, 1999]), this limit access to the authorised users [Misra et al., 2004]. The equation which generates the P-code is well known and unclassified, while the equation which generates the Y-code is classified . This code is being transmitted by both L1 (C/A and P-codes) and L2 (only P-code) carriers and is not accessible to unauthorised users of GPS.

2.3 The GPS receivers and antennas

There are receivers for way point navigation, military, civillians; receivers that use the C/A code and those that use P-code; single and dual frequency receivers; and handheld receivers and others more substantial in size. Although different in their design, construction and capabilities, all GPS receivers share common basic functions. These functions are:

- to capture RF signals transmitted by the satellites.
- to separate the signals from satellites in view.
- to perform measurements of signal transit time and Doppler shift.
- to decode the navigation message to determine the satellite position, velocity, and clock parameters.
- to estimate the user position, velocity, and time.

A simple receiving antenna is a device used to convert energy in a time varying electromagnetic wave into electric current to be handled by the electronics in the receiver [Gibbons, 1999]. Given a recent almanac and rough idea of the user locating the receiver determines which satellites are in view. Also given the satellite ID, the receiver knows the structure of C/A code transmitted by it, and attempts to "tune" to it to acquire the signal (track the changes in it continuously).

The antenna may need to operate at just L1 frequency or at both L1 and L2 frequencies, however my focus will be on L1, because of its availability to civil users. Because GPS signals are circularly polarized, all GPS antennas must be circularly polarized as well. To acquire a signal, the receiver generates a replica of the known C/A code, and attempts to align it with the incoming code by sliding the replica in time and computing the correlation. Direct acquisition of P-code is difficult by design due to the length of the code [Misra et al., 2004].

Below I will discuss the most commonly used GPS receivers, specially designed for ocean and land remote sensing.

2.3.1 The Delay Mapping Receiver (DMR)

Most GPS reflection experiments to date have been conducted with the DMR designed by Dr. Garrison and Dr. Katzberg from Goddard Space Flight and Langley Research Centres. The system is based on the GEC Plessey GPSBuilder-2 [Garrison et al., 2002] (a reference thereof). The receiver includes two low gain L-band antennas, a zenith mounted right-hand circular polarized (RHCP) antenna and a nadir mounted left-hand circular polarized (LHCP) antenna. The zenith-oriented RHCP antenna is used to track the direct line of sight satellites signals, while the nadir-oriented LHCP antenna

tracks the reflected signal (refer figure 3.1).

The two modes operation for the receiver were defined as serial and parallel [Garrison et al., 2002]. The Serial Delay Mapping Receiver (SDMR) tracks up to six satellites and generates their pseudoranges. This position is used to initialize the code delay and Doppler frequency in the reflected channels. The Parallel Delay Mapping Receiver (PDMR) continuously records the cross-correlation in 10 to 12 range bins at fixed delays, all from one or two satellites. The key features are that some of the 12 receiver channels (usually 4-6) are operated in normal closed loop configurations using an upward looking RHCP antenna. The remainder of the channels run open loop using a downward looking LHCP in order to measure the code cross-correlation at a variety of delays.

2.3.2 Digital Beam-Steering GPS receiver

"The NAVSYS High-gain Advanced GPS Receiver (HAGR) is a digital beam steering receiver designed for GPS satellite radio navigation and other spread spectrum applications" [Stolk et al., 2003]. The HAGR is available for both military and commercial precision GPS applications. This can track up to 12 satellites simultaneously. In the normal mode of operation, the beams follow the satellites as they move across the sky. For bistatic signal processing, the beams can be directed at any particular point of interest on the earth. The [Stolk et al., 2003] paper demostrate the ability of the HAGR receiver to improve the GPS bistatic remote capability by using a Digital Beam-Steered to allow weak GPS signal returns to be detected.

2.4 Summary

The advantages of using the GPS signal for remote sensing over the traditionally used remote sensing tools has attracted more researchers to use it. The future inclusion of Galileo in GNSS system will complement the system in terms of its coverage and life time, since most of the GLONASS satellites are dying out. Most researchers (eg. [Komjathy et al., 1998]) compare their results with the TOPEX altimeters, and most wind speed results indicates 2 m/s agreement. From the above statement, it is clear that the GPS remote sensing tool is here to stay.

The availability of the primary L1 frequency carrier to civil users makes the system less complex and more flexible. However those who have access for both L1 and L2 carriers will even get more accurate measurements, and will also be able to calibrate the ionospheric refraction. With the HAGR receiver we can get better coverage compared to DMR.

Having covered the GPS signal structure and receivers, we can find out what is happening on the earth's surface and be able to come up with a 3D reflection geometry. More of this will be unpacked in the following chapter. But I will start by introducing the bistatic geometry before concentrating on the GPS satellite/receiver footprint, because that is where the information about the reflection can be unravelled.

Chapter 3

Geometrical Theory

This chapter will cover the theory pertaining to the geometrical models. Firstly, I will discuss the bistatic geometry of the GPS signal and the models used in many softwares and by receivers to date. The use of GPS in a bistatic radar configuration to measure surface properties relies upon the ability to extract information from the signal. Therefore it is important to discuss the bistatic geometry in detail. It is also interesting to discuss the multipath, since the signals received can be from different GPS satellites and reflection points. The multipath discussion arises questions about resolution and the area of reflection, and the answers will be covered by the Fresnel theory discussion and the models pertaining to it. The models entail the 3D-reflection. Also we will discuss the effect of the Earth's curvature on the model.

3.1 GPS bistatic geometry

Bistatic radars are systems in which spatial separation exists between the transmitting and receiving parts. The bistatic model discussed below was developed by Zavorotny and Voronovich (also known as Z-V model) and is extensively documented in [Zavorotny et.al., 2000]. The Z-V model employs

a forwardscatter radar equation with the geometric optics limit of the Kirchhoff approximation [Komjathy et.al., 2001]. "The Kirchhoff approximation implies that we use a "smooth" rough surface that can be approximated with a tangent plane at any point on the surface" [Komjanthy et.al., 2000]. It is used to predict the power distribution of the reflected signal as a function of time delay. The Z-V model takes the form:

$$\langle \mid Y(\tau, D) \mid^2 \rangle = T_i^2 \int \frac{\Re^2 D^2 \Lambda^2 [\tau - (R_0 + R)/c]}{4R_0^2(\vec{\rho}) R^2(\vec{\rho}) q_z^4} \times \mid S[f_D(\rho) - f_c] \mid^2 P\Big(-\frac{\vec{q}_\perp}{q_z}\Big) q^4 d^2\rho \tag{3.1}$$

where,
$\mid Y(\tau, D) \mid^2$ - is the reflected power;
τ - delay time bin (lag);
T_i - is the integration time in seconds;
$\Re$ - is the complex reflectivity of the ocean at L1;
D - is the antenna gain of the receiver;
Λ - is the correlation function of the GPS C/A code;
S - is the Doppler sinc function;
P - is the probability density function (PDF) of the surface slopes;
q - is the magnitude of the scattering vector $\vec{q}$;
R_0 - is the distance from some point on the surface point to the GPS satellite;
R - is the distance from the GPS receiver to some point on the surface;
c - is the speed of light;
f_d - is the Doppler shift at the specular point;
f_c - is the compensation frequency or the Doppler offset to some point $\vec{\rho}$; and
$\vec{\rho}$ - is a vector from the specular point to some other point on the surface.

GPS scattering geometry is shown in Figure 3.1. As the figure illustrates, the signal reflected from the ocean surface originates from a glistening zone

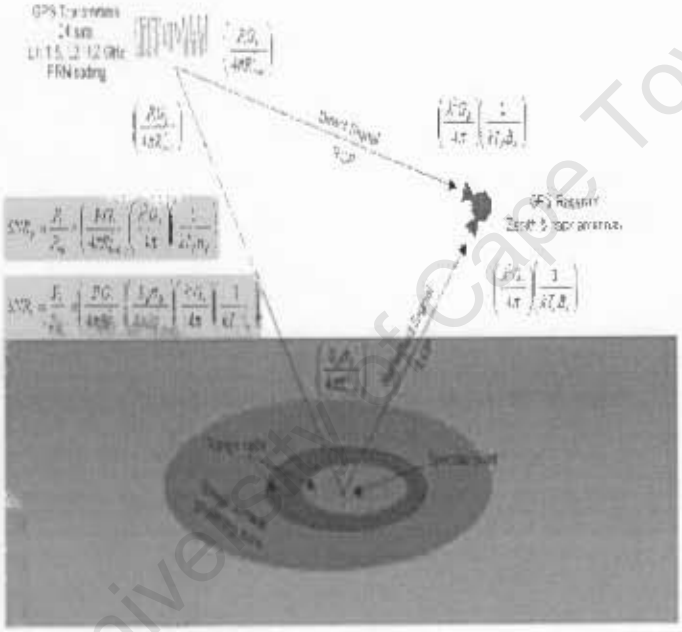

Figure 3.1: Geometry of the ocean-reflected signal.

surrounding nominal specular point. The size and shape of the glistening zones depends on the roughness of the ocean surface [Komjathy et.al., 2001]. In order to measure the reflected power from this glistening zone, the receiver-generated PRN codes are delayed in time with respect to directly received, line-of-sight signals, in turn isolating power originating from specular reflection point surrounding region. Figure 3.2 shows the shape of the resulting waveform of power-versus-delay depends on the ocean surface roughness. In turn this roughness is a function of the surface wind speed and direction, and therefore provides a means to retrieve these geophysical parameters from GPS reflected signal power [Komjathy et.al., 2001].

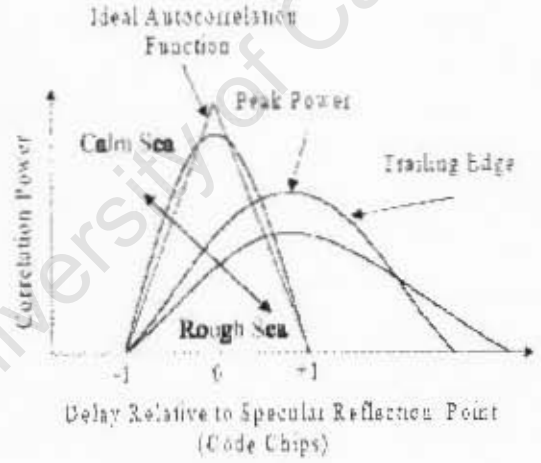

Figure 3.2: Correlation functions shapes for ideal direct GPS signal and for reflected signals from rough surfaces (copied from [Komjathy et.al.,2000]).

The GPS signal surface-scattering from the ocean surface is similar to the land surface, but potentially more complex due to heterogeneous reflection surfaces [Masters et.al., 2001]. The main difference is in the spatially and temporally dielectric constant, surface roughness and possible vegetation cover. Most land reflections originate from smooth surfaces so that they resemble calm seas reflections. The Z-V model is based upon physical optics and assumes a rough surface on flat Earth.

3.1.1 Bistatic GPS signals

There are two types of bistatic GPS signals, diffuse and specular. An optical geometry characterises the specular bistatic GPS signals, providing a very powerful reflection. The strength of the specular returns makes them easy to detect. However it does have some drawbacks, such as the limited area of the earth's surface they provide information on. The smoothness of the surface appears to be a dependent factor of the strength of the specular return. Water provides much stronger specular than uneven surfaces such as forests.

Diffuse bistatic returns are produced by the scattering of the GPS signals on the earth's surface. The diffuse returning signal originate from a much larger area which could potentially provide information over much larger region than possible by processing just the specular returns. However the drawback is that these signals are extremely weak and require advanced receiver technology in order to be useful in remote sensing applications.

At this stage the smoothness and the roughness are unclear and raising the following question: what do we usually mean by a "smooth" surface? The following section attempts to answer the question.

3.2 Smooth or Rough-Surface Criteria

Rayleigh quantifies the roughness of the surface through a simple expression: the Rayleigh criterion, which states that "if the phase difference $\Delta\phi$ (due to propagation) between the two reflected rays shown is less than $\pi/2$ radians, then the surface maybe considered smooth" [Ulaby et.al., 1982]. Rayleigh suggested a way of formulating the relation involving the parameters in figure 3.3 in the following: consider the two rays on a surface with irregularities of height Δh at a grazing angle θ. The path difference between the two rays is given by,

$$\Delta r = 2\Delta h \sin\theta \tag{3.2}$$

and hence the phase difference is,

$$\Delta\phi = \frac{2\pi}{\lambda}\Delta r = \frac{4\pi\Delta h}{\lambda}\sin\theta \tag{3.3}$$

For small values of $\Delta\phi$ the two rays will be almost in phase, as in the case of a perfectly smooth surface. If the phase difference increases until $\Delta\phi = \pi$ where they will be in phase opposition and cancel. The value half-way between the two extremes (rough and smooth) can be used to divide the two cases i.e.$\Delta\phi = \pi/2$ and substituding back to equation 3.3 to get the Rayleigh criterion. According to the criterion the surface is smooth when,

$$\Delta h < \frac{\lambda}{8sin\theta} \tag{3.4}$$

where,

- Δh - is the mean height of irregularities within First Fresnel ellipse.
- λ - is the signal wavelength.
- θ - is the grazing angle or elevation angle of the signal.

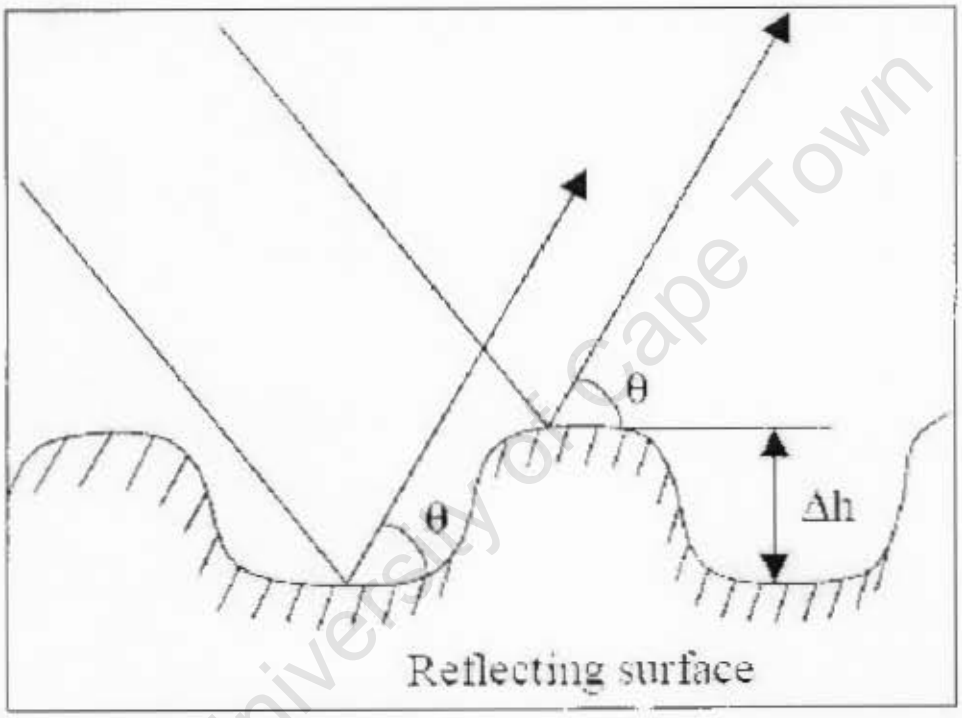

Figure 3.3: Electromagnetic reflection from surface(adapted from [Ray, 2000])

The factor $\frac{1}{8}$ can be switched to either $\frac{1}{16}$ or $\frac{1}{32}$ (Fraunhofer criterion) to characterised different cases of smoothness[Ulaby et.al., 1982]. When the vertical scales of the roughness are greater than GPS L1 wavelength λ, we expect to find other further delayed contribution from multipath of the signal onto the roughness features of the surface [Cardellach et.al., 2002]. Since the signal is not reflected from one point, it is vital to discuss the multipath briefly.

3.3 Multipath Theory

Multipath is the phenomenon whereby a signal arrives at a receiver via multiple paths attributable to reflection and diffraction [Ray, 2000] (a reference thereof). And multipath is a major source of error in GPS code and carrier phase measurements in the differential mode of operation which can prevent the highest level of accuracy, refer the [Ray, 2000] for more information about its effect on the code. Today the phenomenon used to be considered the error source, revolutionalised the new way of remote sensing. By measuring the path delay time of the GPS signal we will be able to determine the most vital oceanographic and metereological data. The following section will show how do we determine the path delay time, which is related to the path difference.

3.4 Fresnel Theory

Having explained the glistening zone and the kind of signals reflected on it, the next point to tackle is which scattering regions are the most important in contributing to the total field at a receiving point (GPS receiver) when the surface is illuminated by the GPS signals at a given point. "The first few Fresnel zones" is the widely accepted answer.

Signal diffration effects can be described by Fresnel zones where a Fresnel zone is the volume of space enclosed by an ellipsoid which has a transmitter and receiver at its foci as shown by figure 3.4. The First Fresnel zone is characterized by a phase difference of $\lambda/2$ with respect to line-of-sight. If the phase difference is increased in steps of $\lambda/2$, a family of ellipses on the plane will result.

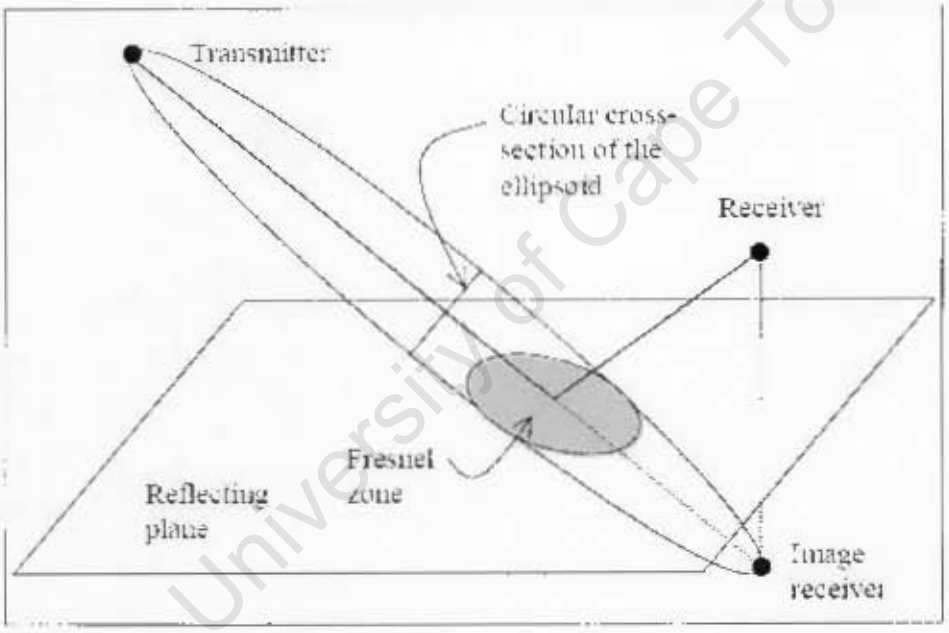

Figure 3.4: Fresnel zone on a reflecting plane(Adapted from Ray, 2000])

To visualize this, consider Figure 3.5 which illustrate the geometry [1] of GPS reflection surface reflection scheme. To explain this in terms of phase

[1] The geometry is considered 3D because the positions are the functions of X,Y and Z.

shifts, let the first phase denote the line-of-sight (LOS) which is given by,

$$\Phi_1 = k \mid \mathbf{RG} \mid \tag{3.5}$$

and the other phase called the scatter path phase shift, given by,

$$\Phi_2 = k\,(\mid \mathbf{RS} \mid + \mid \mathbf{SG} \mid) \tag{3.6}$$

whereby,

$\mid \mathbf{RG} \mid$ - is the distance between the GPS receiver and the GPS satellite,
$\mid \mathbf{RS} \mid$ - is the distance from the GPS receiver to the reflecting surface point,
$\mid \mathbf{SG} \mid$ - is the distance from the reflecting surface point to the GPS satellite, and
$k = \frac{2\pi}{\lambda}$ - is the propagation constant.

Finally the phase difference is given by,

$$\Delta\Phi = \Phi_2 - \Phi_1. \tag{3.7}$$

For the direct path phase to differ from the reflected path phase by an integer multiple of π the paths must differ by integer multiples of $n\lambda/2$. The collection of points at which reflection would produce an excess path length of $n\lambda/2$ is called the n^{th} Fresnel zone. In terms of phase shift, the n^{th} Fresnel zone is denoted by $\Delta\Phi = \Delta\Phi_0 + n\pi$. Since successive zones are in phase opposition, the contribution of adjacent zones will tend to cancel. However, as the excitation amplitude decreases slowly from zone to zone, it leaves the total radiation of the area of approximately half the first Fresnel zone [Beckmann et.al., 1963]. As a results the first Fresnel zones makes the most important contribution to the total field received at **R** (refer Figure 3.5). Refer [Beckmann et.al., 1963] and [Klukas et.al., 2004] for more information

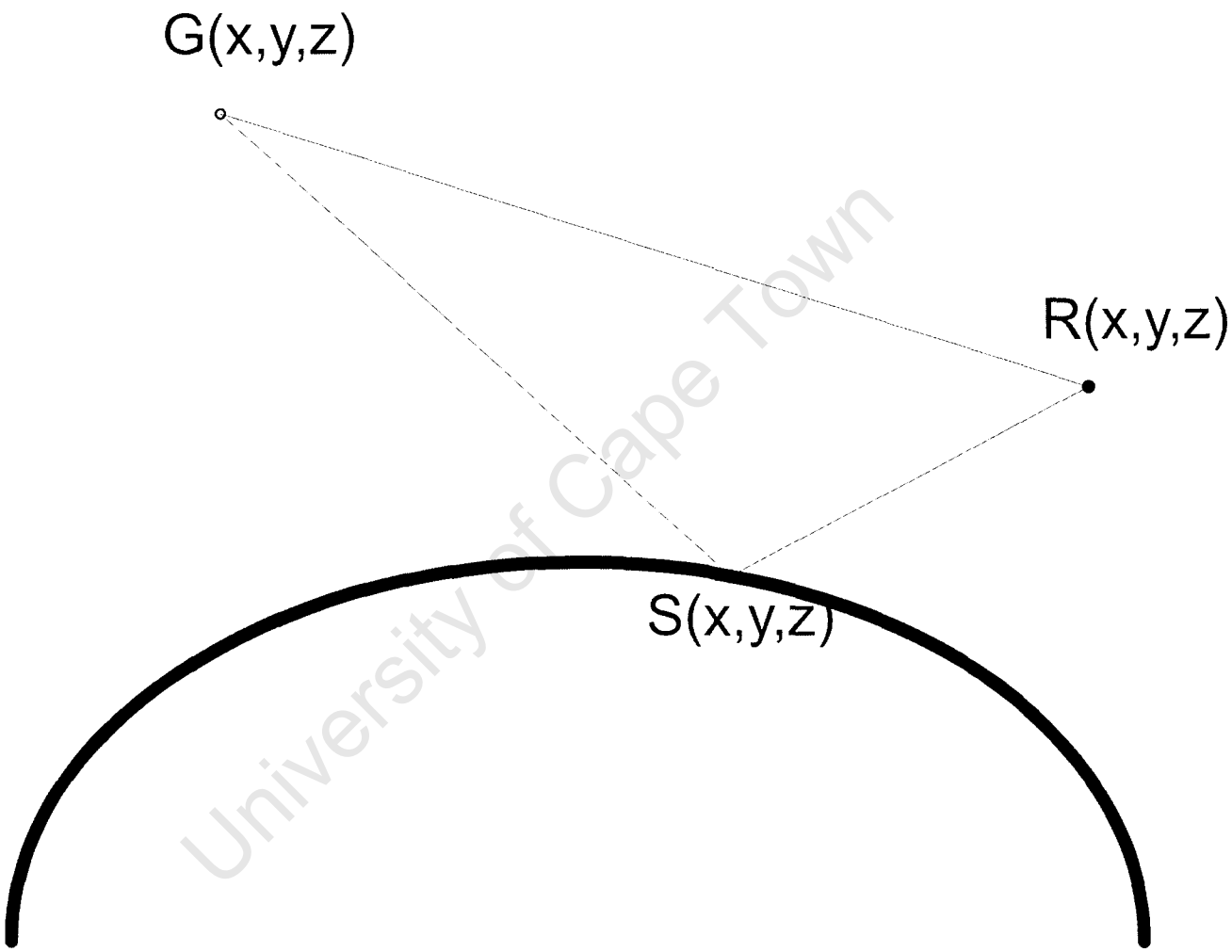

Figure 3.5: Illustration for the 3D Geometry of GPS Surface Reflection Scheme.

on Fresnel zone theory.

The other important factor that quantifies the Fresnel zones is the Fresnel zone radius, and is given by [Ray, 2000]:

$$F_n = \sqrt{\frac{n\lambda \mid \mathbf{RS} \mid\mid \mathbf{SG} \mid}{\mid \mathbf{RS} \mid + \mid \mathbf{SG} \mid}} \tag{3.8}$$

If $\mid \mathbf{SG} \mid$ is much greater than $\mid \mathbf{RS} \mid$, then:

$$F_n = \sqrt{n\lambda \mid \mathbf{RS} \mid}. \tag{3.9}$$

Lastly, the footprint of the ellipsoid on the plane of reflection is of the form of an ellipse, with a semi-major axis approximated to:

$$a = \frac{F_n}{sin\theta}. \tag{3.10}$$

From Equation 3.10, it is clear that for a low elevation angle θ the semi-major axis is large. For a small elevation angles the ellipses are very prolonged, getting longer and narrower with decreasing elevation angle (refer [Beckmann et.al., 1963] pp. 13 table for numerical examples). As a result, the smaller the elevation angle, the larger the semi-major axis and the area of the ellipse.

The Fresnel zone concept has turned many heads in wireless camps, and its application has been extended to designing what is called the Fresnel zone plates [Bricchi et.al., 2002]. The combination of both the multipath and Fresnel zones also have an application in wireless ad hoc networks. "Fresnel zone routing" (FZR) constructs multiple parallel paths from source to destination based on the concept of "Fresnel zones in a wireless network (refer to [Liang, 2004] for more information).

3.5 Effect by Earth's curvature

The remaining question is how and when does the earth's curvature affect measurements. The simple answer to this is when the transmitter to receiver distance becomes too large. When the transmitter and the receiver are placed far from each other the line-of-sight (LOS) gets shadowed by the earth's surface, and as a result cannot be called a line-of-sight any more. Similiar situations are discussed in most papers as radio occultation (refer [Mortensen et.al., 1998] for occultation geometry). The solution to this problem is to lift the receiver higher, more explanation can be found on ZyTrax [2] website.

3.6 Summary

The chapter has covered the depth of the reflection model, from bistatic geometry to Fresnel zones. The parameters that quantify the Fresnel zones have also been covered in the chapter. Based on the discussion on this chapter, the challenge is to demonstrate if the models do work through the implementation of the software written in IDL code. This leads to the next chapter which defined the most important physical constants, equations and algorithms used to write the software. Chapter three concludes the end of literature review and the next chapter discusses practical applications.

[2] http://www.zytrax.com/tech/wireless/fresnel.htm

Chapter 4

Global Coordinate Systems and Algorithms

This chapter defines the global coordinate systems (GCS) designed to represent the position and velocity of the GPS satellite orbiting the earth, and also the position of a point on the earth. The fact that the earth is not a rigid uniform sphere and that its density is not uniform either, does complicate matters. The main aim is to define Cartesian coordinate system fixed to earth and another fixed in orientation relative to the so called 'fixed' stars. I will also show how to transform from one set of coordinates to another. All of the above are done with an eye of introducing the World Geodetic System 1984 (WGS 84). Finally all the equations derived and discussed will help in defining the algorithm, which will also be discussed under this section.

4.1 Terrestrial and Inertial Rererence Systems

4.1.1 Conventional Terrestrial Reference System (CTRS)

Figure 4.1 represents the cartesian coordinates system having origin at the centre of the earth, the x-axis coinciding with the intersection of Greenwich meridian and the equatorial plane, and z-axis coinciding with the axis of rotation. If the complications can be dealt with, this would seem to be a good example for an earth-fixed coordinate system. The polar motion phenomenon occurs when the pole of rotation wanders around the surface of the earth, which is roughly a circular path. Geodesists have defined an average position of the earth's pole of rotation between 1900 to 1905, in order to get around the difficulty of the wandering pole [Misra et. al., 2004]. The fixed to the earth's crust is referred to Coventional Terrestrial Pole (CTP).

The cartesian coordinate system fixed to the earth can be defined as follows:

- centre of the earth's mass is considered the origin.
- z-axis through the CTP.
- x-axis passing through the intersection of the CTP's equatorial plane and a reference meridian.
- y-axis complete the right handed coordinates system (defined in the equatorial plane).

Finally this forms part of the earth-centered, earth-fixed (ECEF) cartesian coordinate system and is referred to as Convertional Terrestrial Reference System (CTRS). The CTRS is realised through the coordinates of a set of points on the earth.

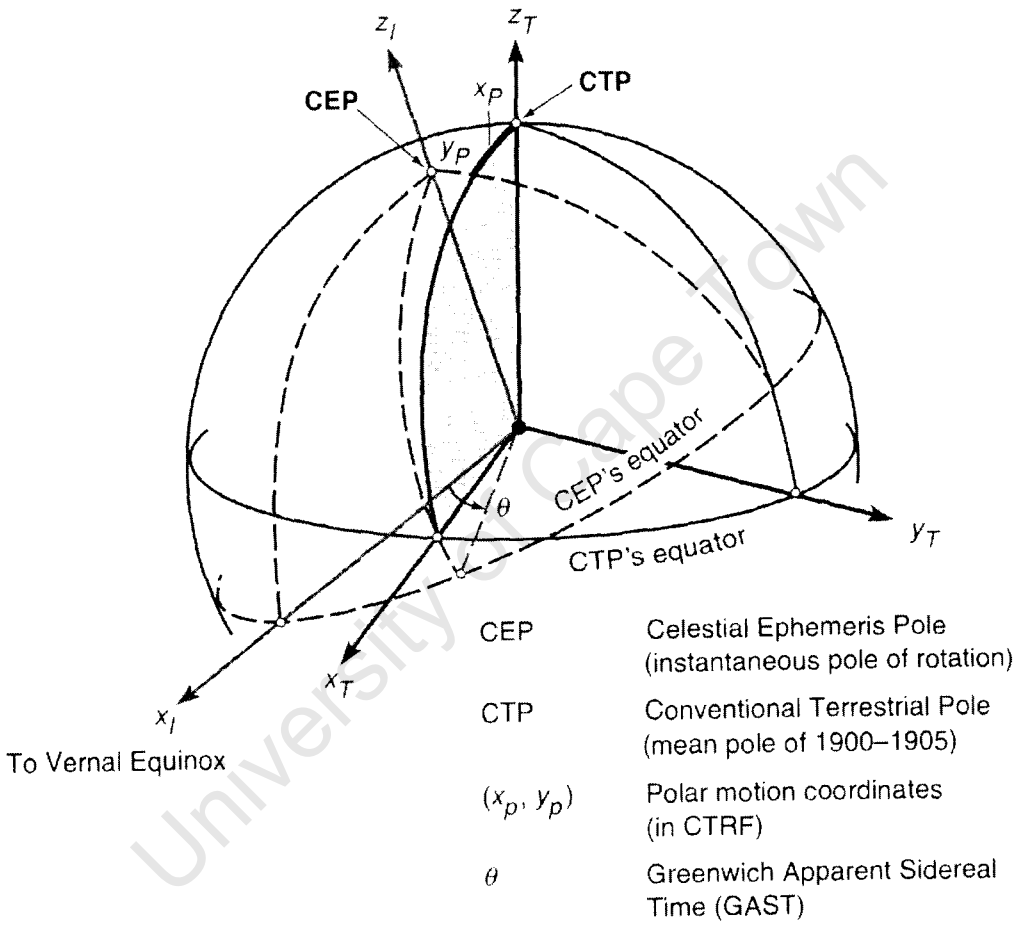

Figure 4.1: Terrestrial and inertial reference systems (adopted from [Misra et.al., 2004])

4.1.2 Conventional Inertial Reference System (CIRS)

The inertial reference system is used for the description of the satellite motion where Newton's law of motions are valid. The inertial system can be defined to be stationary in space, or moving with a constant velocity. The previous discussion of the CTRS, the orientation of the earth in space did not matter. The CIRS considers the fact that the axis of rotation of the earth is not fixed in space in relation to the extra terrestrial objects like stars, quasars, planets, or the moon [Seeber, 2003]. The CIRS consider the periodic components referred to as precession and nutation, in which the motion of the earth's spin axis is space composite. The CIRS is realised through a catalogue of position and proper motions of a set of fundamental stars and Very Long Baseline Interferometry (VLBI) observations of quasars and other extra-galactic objects [Misra et. al., 2004].

An inertial coordinate system can be defined as follows:

- centre of the earth's mass is considered the origin.
- z-axis along rotational axis.
- x-axis in the equatorial plane pointing towards the vernal equinox.
- y-axis complete the right handed coordinates system.

4.2 Geodetic Coordinates, Geoid, and Datums

Discussion of the cartesian coordinates from the above sections pose some limitations in everyday use. An alternative way in mapping the surface of the earth, is to limit the information to a horizontal position only. The horizontal position can be expressed as angular coordinates - longitude and latitude. However the irregularity and changeability of the earth's surface does complicate things as well. Usually a rotational ellipsoid is selected which is flattened

at the poles and created by rotating the meridian ellipse about its minor axis.

4.2.1 Ellipsoidal Coordinates

An ellipsoid is defined in conjuction with an ECEF cartesian coordinate system, with an origin at the centre of mass of the earth. In addition to the definition, there are two important parameters that fully characterise the ellipsoid. These parameters are lengths of semi-major and semi-minor axes, denoted as a and b respectively. The axis of revolution of the ellipsoid coincident with the z-axis. Now the eccentricity is defines as $e^2 = (a^2 - b^2)/a^2$. The eccentricity can as well be defined as $e^2 = 2f - f^2$, where the flattening f is defined as $f = (a - b)/a$.

The ellipsoidal coordinates (sometimes called the geodetic coordinates) with reference to point P (figure 4.2), can be defined as:

- ellipsoidal latitude (ϕ): the angle between the equatorial (x-y) plane of the ellipsoid and the line to the point P (measured positive north from the equator, negative south)
- ellipsoidal longitude (λ): the angle measured in the equatorial plane (measured positive east from the zero meridian)
- ellipsoidal height (h): measured along the normal to the ellipsoid through P.

A geocentric ellipsoid specifies a global datum or a reference surface to be used in defining 3-D coordinates of a point anywhere. The ellipsoidal coordinate system are prefered because they closely approximate Earth's surface [Seeber, 2003], and they facilitate a separation of horizontal position and height.

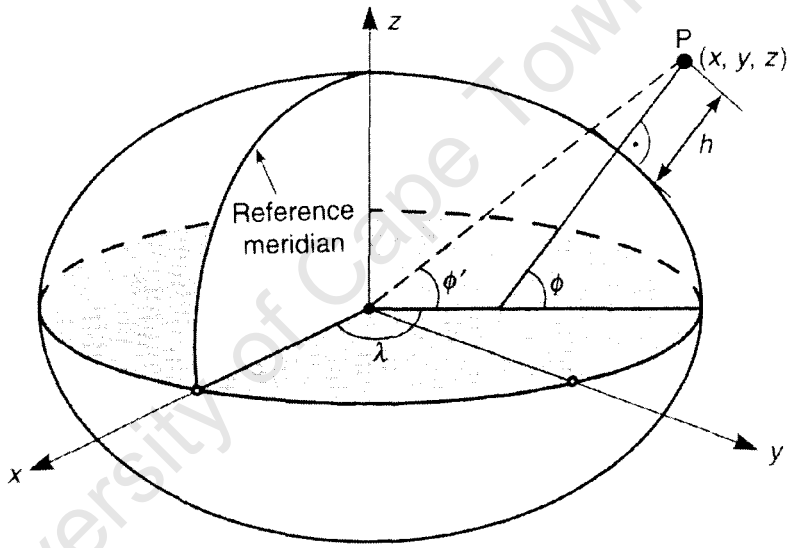

Figure 4.2: Cartesian and ellipsoidal coordinates (adopted from [Misra et.al., 2004])

Conversion between Geodetic and Cartesian Coordinates

The conversion from geodetic (ϕ,λ,h) to cartesion (x,y,z) coordinates is given by the well known equations ([Seeber, 2003], [Misra et.al., 2004] and [Soler, 1998]):

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} (N+h)\cos\phi\cos\lambda \\ (N+h)\cos\phi\sin\lambda \\ (N(1-e^2)+h)\sin\phi \end{pmatrix} \tag{4.1}$$

where N is the principal radius of the curvature along prime vertical [Soler, 1998] and it is given by:

$$N = \frac{a}{\sqrt{1-(e\sin\phi)^2}}. \tag{4.2}$$

It is also important to note the following information about the Figure 4.2,

- origin at the centre of the ellipsoid O
- z-axis directed to the nothern ellipsoidal pole
- x-axis directed to the ellipsoid zero meridian
- y-axis completing the right-handed system.

The inverse transformation (cartesian to geodetic) does not have the simple explicit closed formulation and needs to be solved iteratively. However it can be summarised as follows:

$$\begin{pmatrix} \phi \\ \lambda \\ h \end{pmatrix} = \begin{pmatrix} \arctan\{\frac{z}{p}\left(1-e^2\frac{N}{N+h}\right)^{-1}\} \\ \arctan(\frac{y}{z}) \\ \frac{p}{\cos\phi} - N \end{pmatrix} \tag{4.3}$$

where $p = \sqrt{x^2 + y^2}$. The above equation and discussion are logged in the files, "convert_latlon_xyz.pro" and "convert_xyz_latlon.pro" (refer the appendix).

4.2.2 World Geodetic System 1984 (WGS 84)

WGS 84 is the system that has been developed and maintained by the U.S. Department of Defence (DoD) since about 1960 [Seeber, 2003]. WGS 84 is the geodetic system for all mapping, charting, navigation, and geodetic products to be used throughout the DoD. WGS 84 is a refinement of the earlier versions of WGS 60, WGS 66 and WGS 72. The main objective of the WGS 84 system has been to compute the operational broadcast ephemeris of Transit Doppler and GPS. The widespread use of GPS is turning WGS 84 from global datum into an international datum [Misra et. al., 2004]. The ephemereides of the GPS satellites are expressed in WGS 84, and I have adopted the WGS 84 in all calculations and coding.

Having discussed the WGS 84, it is important to summarise the latest major fundamental constants in the following:

Parameter	**Symbol**	**Value**	**units**
Semi-major	a	6378137.0	m
flattening	f	1/298.257223563	
Earth's angular velocity	ω_E	$7292115.0 \times 10^{-11}$	rad/s
Earth's gravitational constant	GM	398600.4418	km^3/s^2

And the above constants had been logged in a idl procedure file called "define_gps_parameters.pro", refer the appendix for more information.

4.3 GPS Orbits and Satellites Position and Velocity

4.3.1 GPS Orbit parameters

GPS satellite orbits, like any other orbits in space are characterised by six independent parameters (also known as Keplerian elements). They can be summarised as follows:

- a - semi-major axis
- e - numerical eccentricity
- i - orbit inclination
- Ω - right ascension of ascending node
- ω - argument of perigee
- ν - true anomaly.

All six of the elements describe the size and shape of the orbit and its orientation in space ([Misra et.al., 2004] and [Montenbruck et. al., 2000]). The first two parameters (a and e) define the shape of the orbit, while the following three parameters (i, Ω and ω) define the three dimension (3-D) shape of the orbit position. The last parameter (ν) specifies the position of the satellite at a particular time instant. The above parameters are being read from a file called "gps_almanac.txt" using idl procedure file called "get_gps_orbit_parameters.pro".

4.3.2 Satellite Position and Velocity

If the true anomaly of a satellite is given at an epoch, its position can be determined at all times. Following the discussion about the Keplerian elements,

the satellite position and velocity will be expressed in inertial and terrestrial cartesian coordinate system in terms of these elements. As discussed by many literatures ([Misra et. al., 2004], [Seeber, 2003] and [Montenbruck et. al., 2000]), the satellite position can be summarised as follows:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = r \begin{pmatrix} \cos\nu \\ \sin\nu \\ 0 \end{pmatrix} \tag{4.4}$$

where r is called the orbit radius, which is basically the magnitude of the vector and is given by:

$$r = \frac{a(1-e^2)}{1+e\cos\nu}. \tag{4.5}$$

Both the satellite position and the true anomaly are functions of time. The true anomaly can be conveniently obtained in terms of three more parameters. The first parameter is the eccentric anomaly E, by geometrical definition is the "angle subtended at the centre of the orbit between the perigee and the projection of the satellite position on a circle of radius a" [Misra et. al., 2004]. Now the satellite position expressed in terms of eccentric anomaly:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a\cos E - ae \\ a\sqrt{1-e^2}\sin E \\ 0 \end{pmatrix}. \tag{4.6}$$

The eccentric anomaly can be obtained by further introducing the mean motion and mean anomaly. The mean motion, n is given by:

$$n = \frac{\sqrt{GM}}{a^3}. \tag{4.7}$$

The mean anomaly at epoch time t is given by:

$$M = n(t - t_p) \tag{4.8}$$

where t_p is the time of the perigee crossing. Kepler's equation relates the mean and eccentric anomalies as follows:

$$M = E - e \sin E. \tag{4.9}$$

Given M, one can iteratively solve for E using the iterative methods such as Newton's method. The auxillary function can be defined as:

$$f(E) = E - e \sin E - M. \tag{4.10}$$

An approximate root E_i of f may be improved by computing:

$$E_{i+1} = E_i - \frac{f(E_i)}{f'(E_i)}. \tag{4.11}$$

The common way is to start with approximation of $E_0 = M$ (recommended for small eccentricity), and for $e > 0.8$ to start with $E_0 = \pi$ [Montenbruck et. al., 2000].

The satellite velocity vector can be easily obtain by differentianting the satellite position vector once, and can be summarised as follows:

$$\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} = \frac{\sqrt{GMa}}{r} \begin{pmatrix} -sinE \\ \sqrt{1-e^2}\cos E \\ 0 \end{pmatrix}. \tag{4.12}$$

The orientation of these vectors in the earth fixed coordinate system can be found by applying the following rotation matrices:

$$\mathbf{R_x}(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\varphi & \cos\phi \end{pmatrix}$$

$$\mathbf{R_y}(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\varphi \end{pmatrix}$$

$$\mathbf{R_z}(\phi) = \begin{pmatrix} \cos\phi & -\sin\varphi & 0 \\ \sin\phi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

These rotational matrices are logged in the function files called "rotx.pro", "roty.pro" and "rotz.pro" respectively. And the file "transform_inertial_to_fixed.pro" is used to transform a position vector from the inertial system to the Earth-fixed system.

4.4 Other Considerations

As the satellites move, the semi-major axis corresponding to the repeat cycle changes as specified by number of days, and it is modified by the procedure called "get_repeat_cycle_semimajor.pro". The other consideration is to calculate the intersection of the antenna pointing vector with the earth ellipsoid, this is easily done by the procedure called "ellipsoidal_intersection.pro".

4.5 Determining the locations and the size of the active scattering region

The surface point at which the satellite signal is reflected to the satellite receiver is called the reflection point. One has to remember that although

the earth's shape can be thought of as an ellipsoid, the mean surface itself does deviate from the geoid. In a sea surface the deviation can occur due to ocean dynamic currents and earth's tides [Wu et. al., 1997]. Therefore the determination of the reflection points on the earth's surface and the sea becomes one key issue that need to be addressed. I have adopted the algorithm outline by [Wu et. al., 1997], with an assumption of an approximate nominal surface. The initial guess of the reflection point (also called the geocentric position vector **S**) is given by:

$$\mathbf{S} \approx \mathbf{R} + H_r/(H_r + H_g)(\mathbf{G} - \mathbf{R}) \tag{4.13}$$

where the H_r is the height of the GPS receiver and H_g is the height of the GPS satellite. The above equation was used in the idl procedure file called "reflection_pos.pro". Having obtained the reflection point, one had to explore the actual size surrounding these reflection points. To simplify the picture of the Fresnel zone, Figure 4.3 shows the Fresnel zones on a flat reflecting plane.

The equations for these ellipses can be derived using some tedious analytical geometry but they could be summarised as follows (taken from [Beckmann et.al., 1963]) and with semiminor axis given by (in relation to Figure 4.3),

$$y_n = \frac{r}{2}\sqrt{\left\{\left[\left(\frac{\delta}{r}\right)^2 + \frac{2\delta}{r}\sec\theta\right]\left[1 - \frac{\tan^2\theta}{(\delta/r + \sec\theta)^2 - 1}\right]\right\}} \tag{4.14}$$

and the semimajor axis,

$$x_n = y_1\sqrt{\left\{1 + \frac{1}{(\delta/r + \sec\theta)^2 - 1}\right\}} \tag{4.15}$$

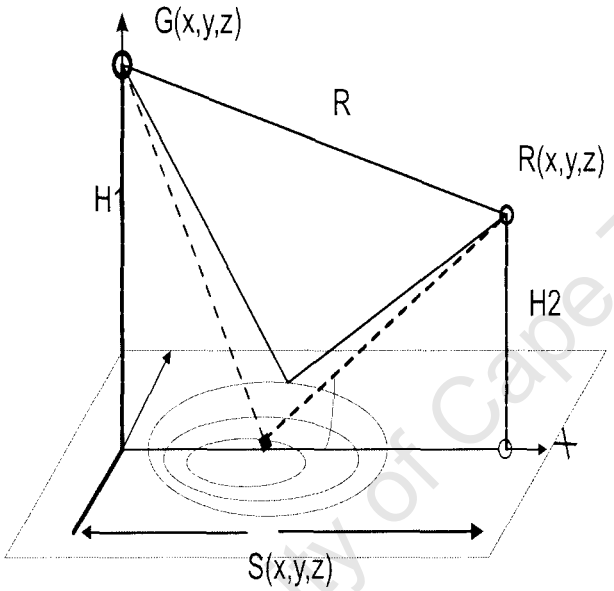

Figure 4.3: The Fresnel zones on a reflecting plane (adopted from [Beckmann et.al, 1963]).

where θ is the slant angle of R, i.e.

$$\tan\theta = \frac{H2 - H1}{r}. \tag{4.16}$$

δ is the path difference used to express the Fresnel zones by setting it to $\lambda/2$ [Kartzberg et.al.,1996], r is the horizontal separation between the GPS satellite $\mathbf{G}(\mathbf{x},\mathbf{y},\mathbf{z})$ position and receiver $\mathbf{R}(\mathbf{x},\mathbf{y},\mathbf{z})$ position. $H1$ and $H2$ represent GPS satellite height and satellite receiver height respectively. All of the above equations are logged in a file called "fresnel_ellipse.pro".

4.6 Summary

Having obtained all the models and equations, the idl procedures were created and interlinked together to simulate some situations (scenarios). The results are obtained and discussed in the following chapter.

Chapter 5

Results and Discussion

The simulations below were modelled for the situation of a moving GPS satellites (using the recent GPS parameters read from the "gps_alamanac.txt" file), land-based receiver and both the pressurised and non-pressurised aircraft mounted receiver. The first thing to look at is the global GPS coverage considering both the orthogonal projection and cylindrical maps presented by Figure 5.1 and 5.2.

Taking a closer look at the Figure 5.1 and Figure 5.2 again, the circle with a cross on it represents the actual position of the satellite. The direction at which each satellite is heading can be determined by tail to head direction. Most of the satellites are visible on the cylindrical projection map, and least satellites are visible on the orthogonal projection map, because the orthogonal projection map is being viewed side way.

Only South African (SA) regions (ocean and land) were considered in these observations. With the help of the idl procedure file "plot_map.pro", the coordinates representing South African can be easily be selected and together with the type of projection map of choice. For convenient reason the cylindrical projection was used throughout, since the satellite subtracks can

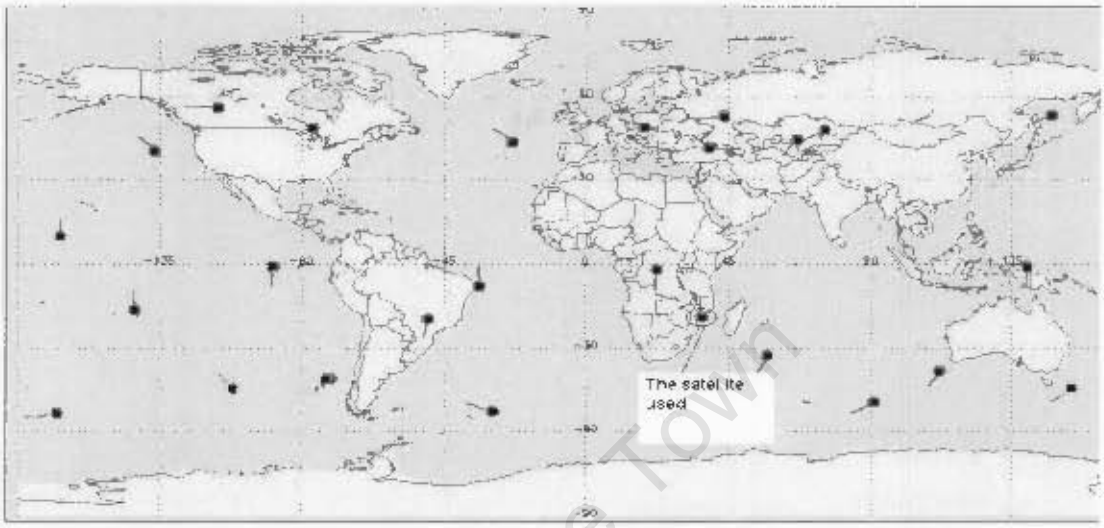

Figure 5.1: Cylindrical projection map showing snap shot of the GPS satellites coverage

be seen as the satellite rises and sets on the horizon. Figure 5.3 shows the snap shot of the Republic of South African region to be considered throughout. Having identified the satellite visible in SA region, the GPS receivers were position in different regions at different heights and the point at which the signal is reflected on the (reflection point) from the satellite to the receiver is observed. Considering one GPS satellite initially the observation are shown by figures 5.4-5.7. The coordinates chosen to place the GPS receivers are the well established coordinates (Durban, East London, George and Table Mountain) in SA at the moment, where by the GPS receivers are being installed there (more on trinet[4] website).

All the receivers represented by figures 5.4-5.7 are being put at the height 1000 m above sea level, except the one placed at Table Mountain at the height of 1100 m. The blob without the tale on the plots always represents

[4] http://www.trinet.gov.za

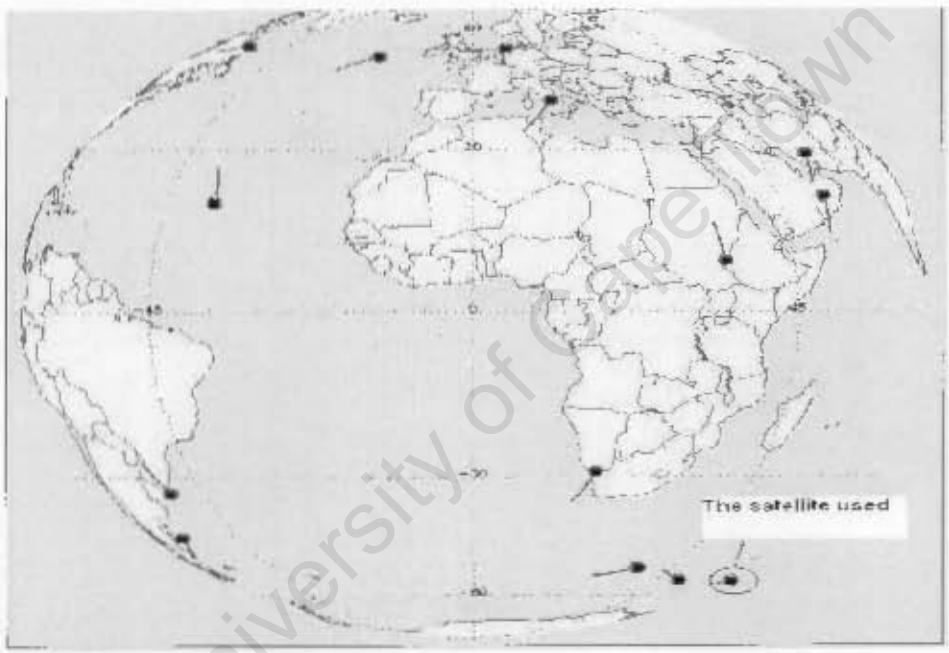

Figure 5.2: Orthogonal projection map showing snap shot of the GPS satellites coverage

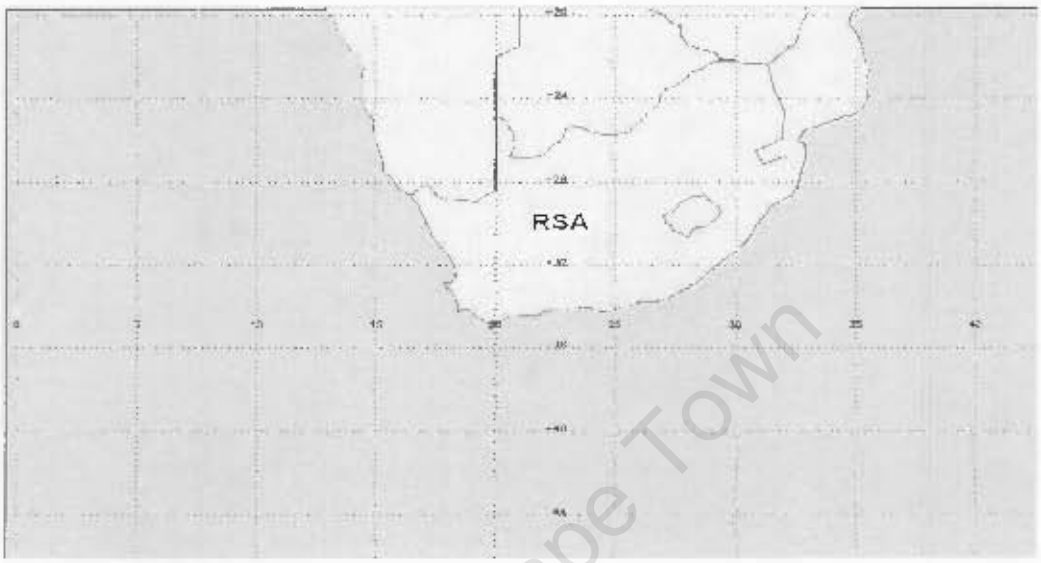

Figure 5.3: Cylindrical projection of the Southern African (RSA) map.

the position of the GPS receiver, while the blob with a shorter tail represents the reflection point (always in between) and the last blob with a larger tail represent the GPS satellite. The above observations were made with the same GPS satellite as a reference point. The reflection point turns to depend on the geographical position of the GPS receiver, and it is also important to note that the satellite passes closer to the GPS receiver placed in Durban as viewed from the top looking downwards.

Figure 5.8 shows that the GPS satellite tracked was visible over the middle of the South Indian Ocean when the simulation started (refer Figure 5.1 for the actual position in the world map). It is also important to note that the satellite passes South Africa at about 145 simulation time steps ($145 \times 500/3600 \approx 20$ hours) onwards. This region of about 145 time steps will be more important in the analysis of the figures below since the GPS receiver used was placed around that region.

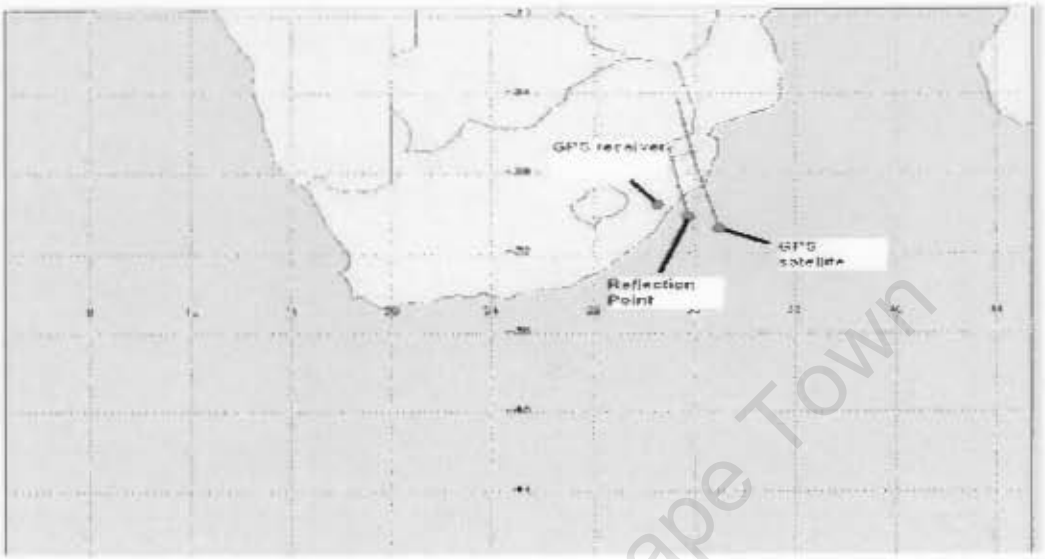

Figure 5.4: Durban GPS receiver placed at [-29.57, 30.56, 1000] coordinates.

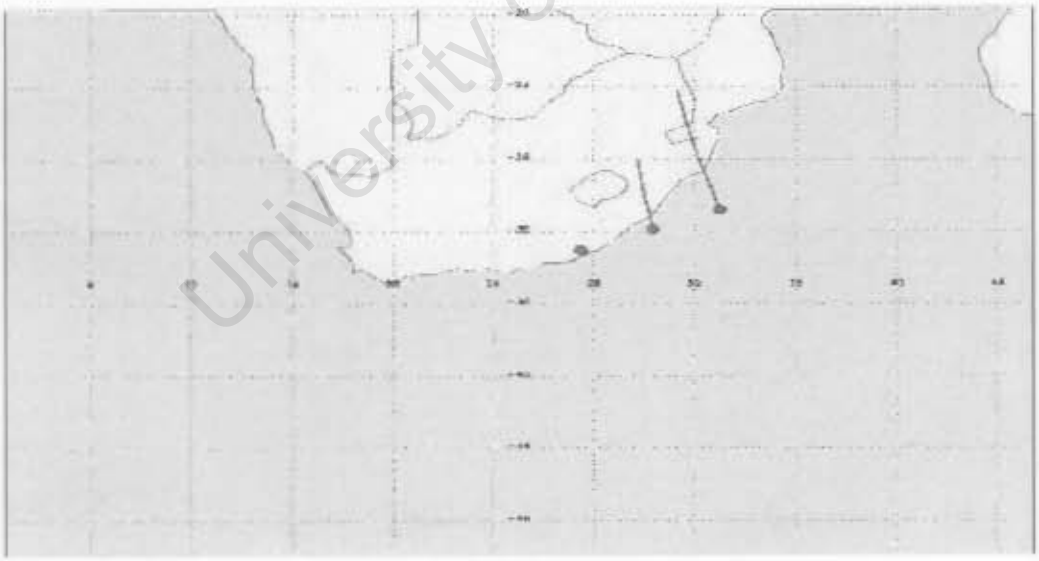

Figure 5.5: East London GPS receiver placed at [-33.02, 27.49, 1000] coordinates.

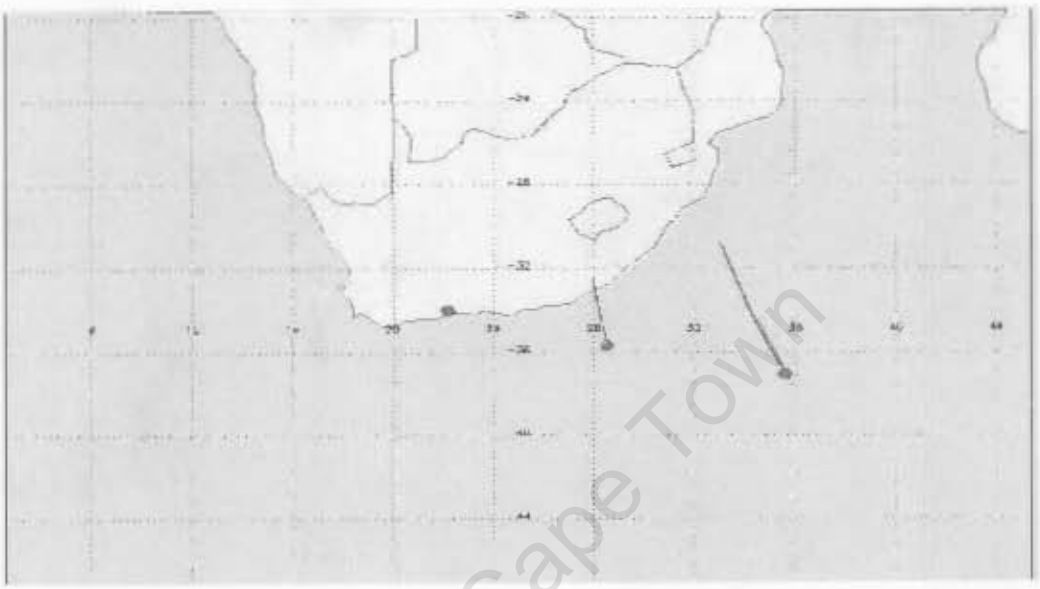

Figure 5.6: George GPS receiver placed at [-34.00, 22.22, 1000] coordinates.

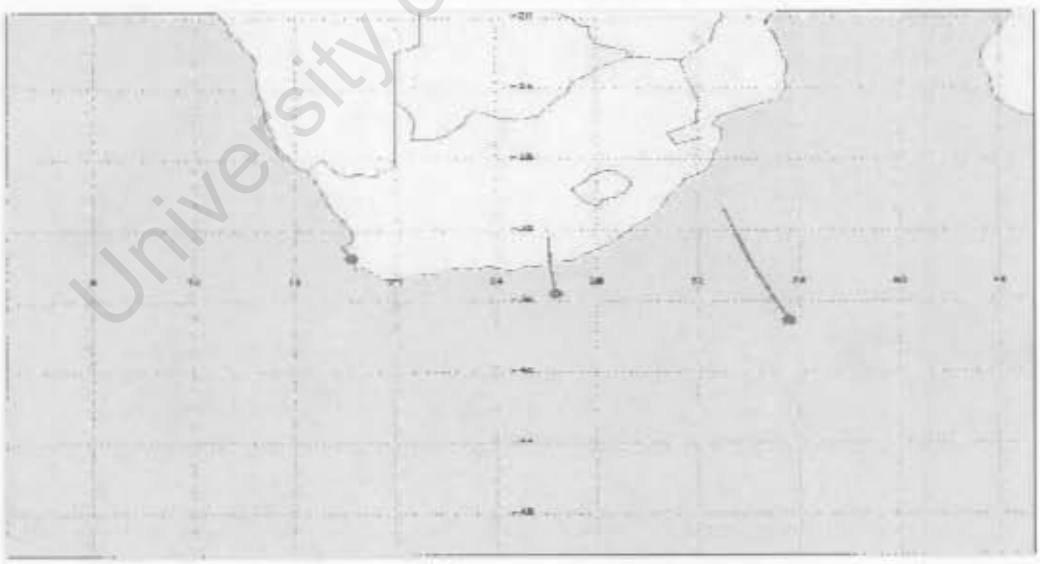

Figure 5.7: Table Mountain GPS receiver placed at [-33.57, 18.28, 1100] coordinates.

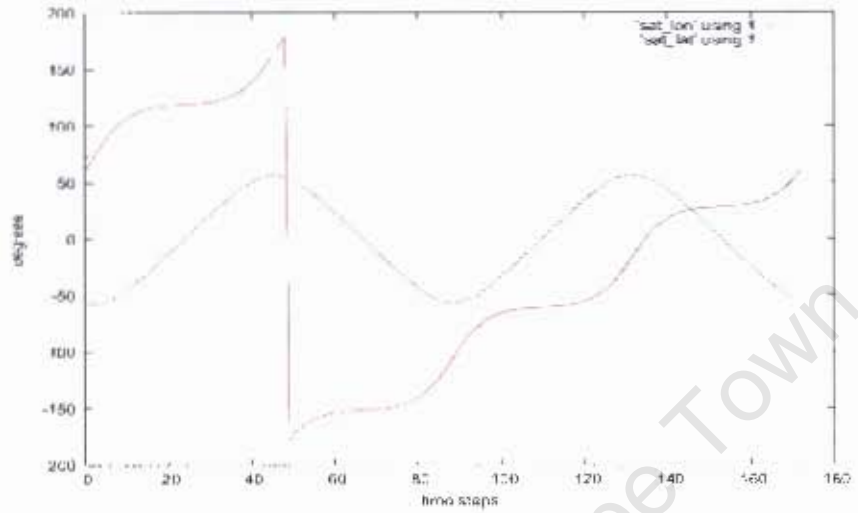

Figure 5.8: GPS satellite lattitude and longitude tracks, ran for 24 hours with a step length of 500 sec.

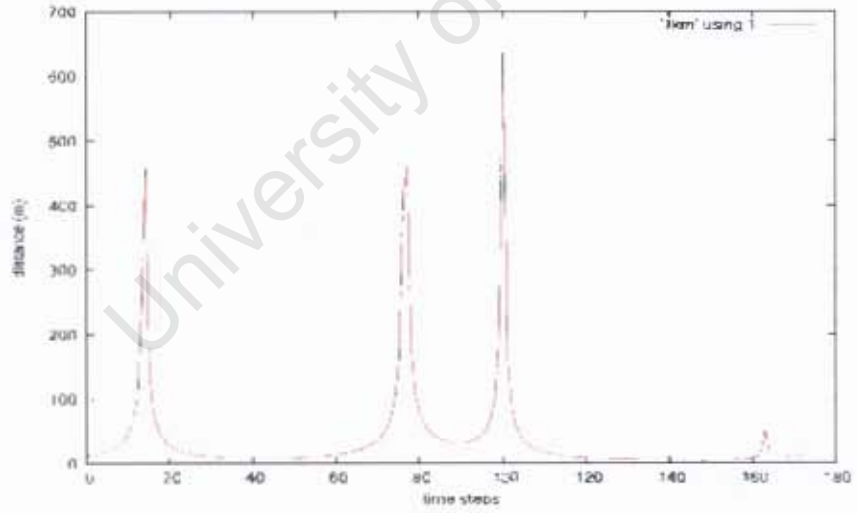

Figure 5.9: Semi-minor of the Fresnel zone (East London receiver placed at 1km height)

At this stage it is important to verify that the size of the fresnel zones

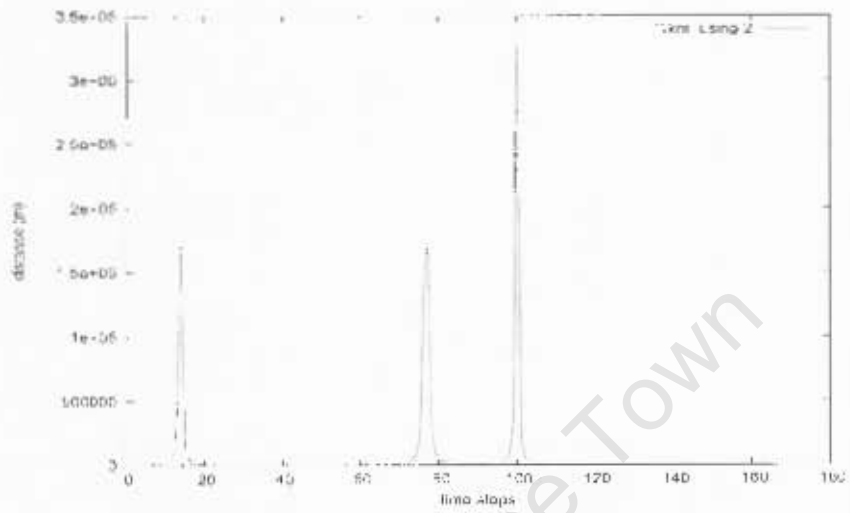

Figure 5.10: Semi-major of the Fresnel zone (East London receiver placed at 1km height)

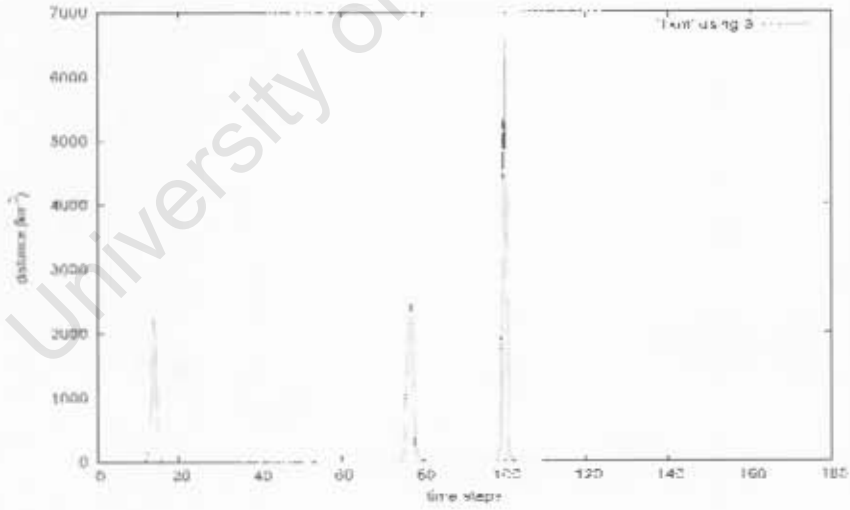

Figure 5.11: Area of the Fresnel zone (East London receiver placed at 1km height)

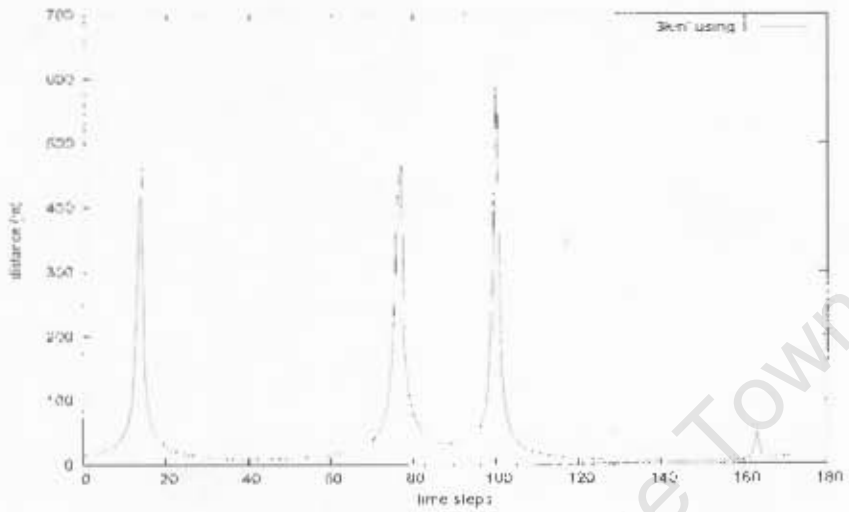

Figure 5.12: Semi-minor of the Fresnel zone (East London receiver placed at 3km height)

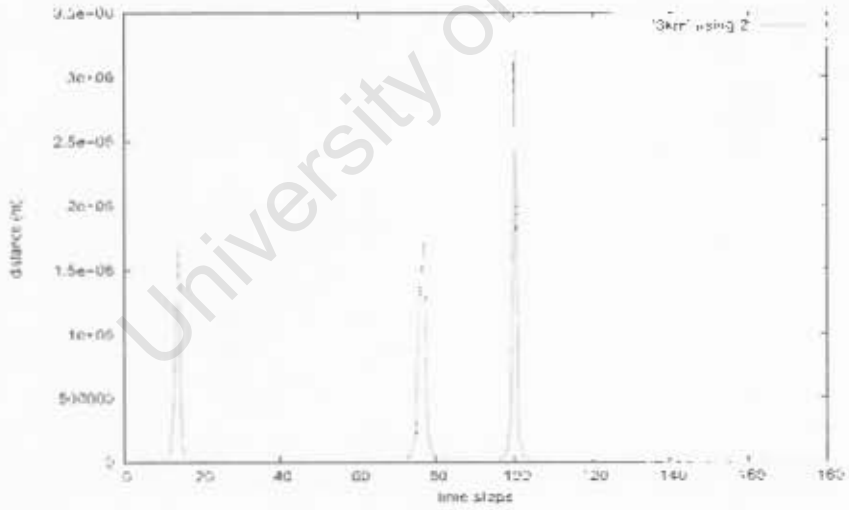

Figure 5.13: Semi-major of the Fresnel zone (East London receiver placed at 3km height)

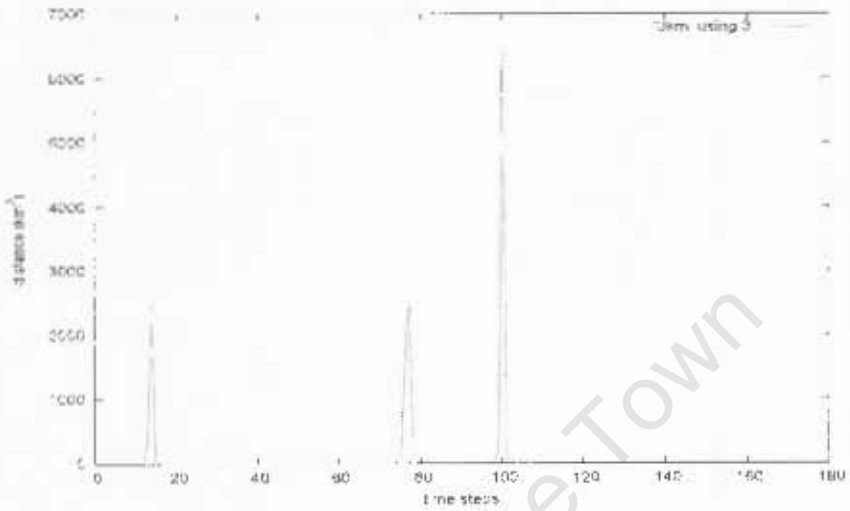

Figure 5.14: Area of the Fresnel zone (East London receiver placed at 3km height)

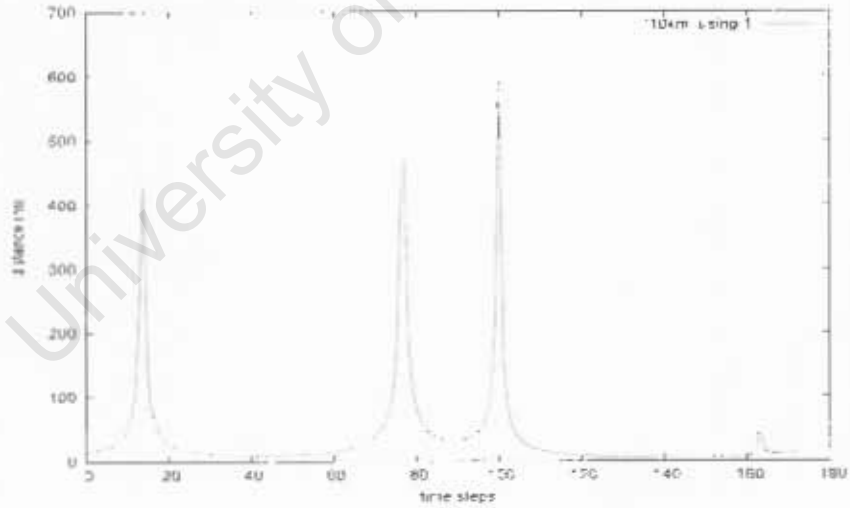

Figure 5.15: Semi-minor of the Fresnel zone (East London receiver placed at 10km height)

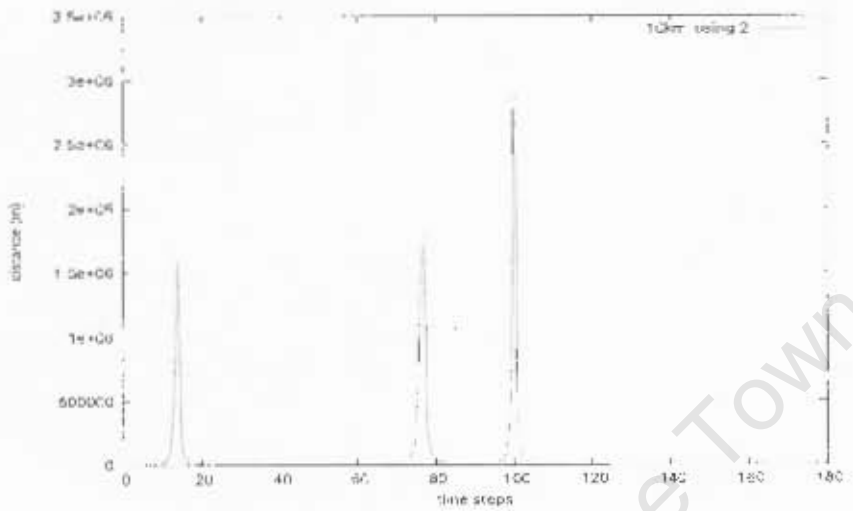

Figure 5.16: Semi-major of the Fresnel zone (East London receiver placed at 10km height)

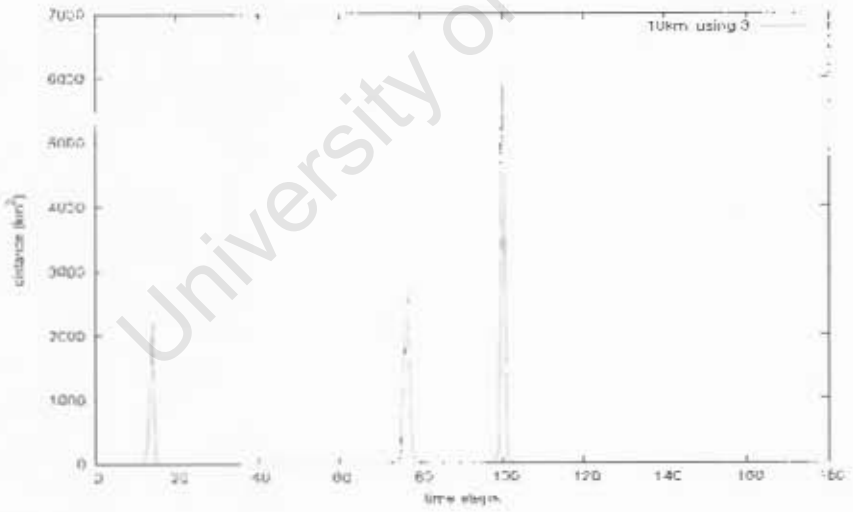

Figure 5.17: Area of the Fresnel zone (East London receiver placed at 10km height)

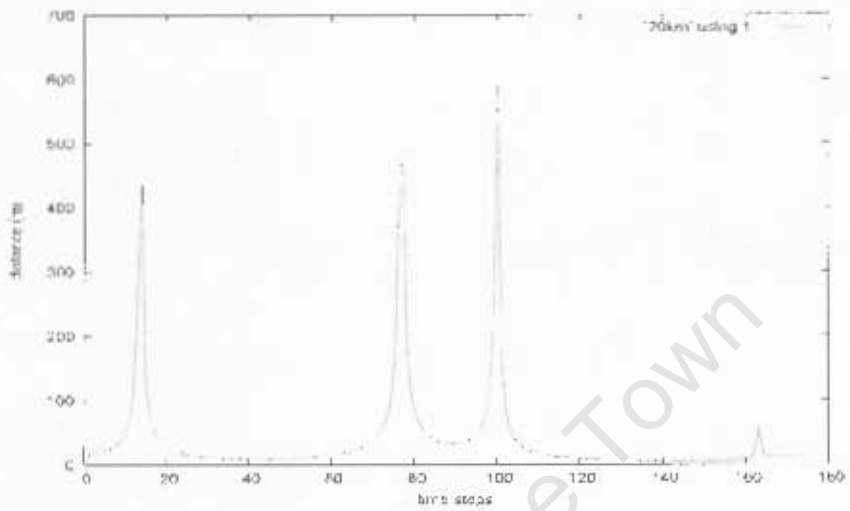

Figure 5.18: Semi-minor of the Fresnel zone (East London receiver placed at 20km height)

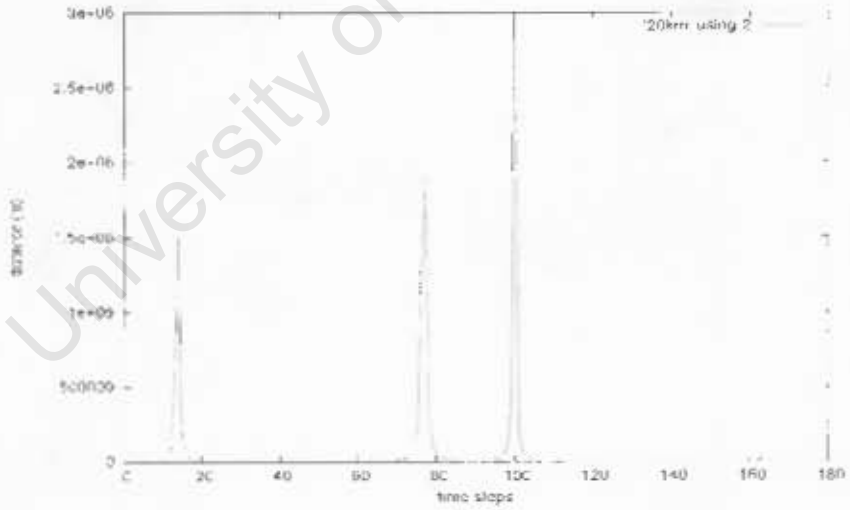

Figure 5.19: Semi-major of the Fresnel zone (East London receiver placed at 20km height)

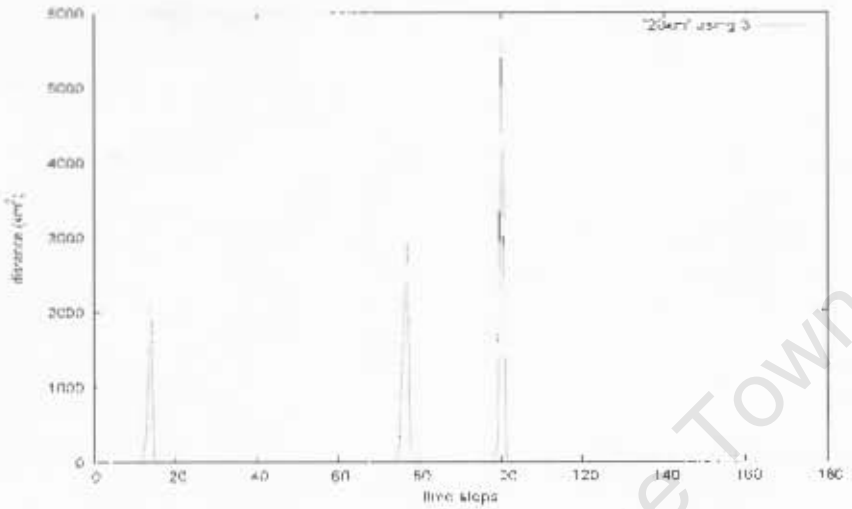

Figure 5.20: Area of the Fresnel zone (East London receiver placed at 20km height)

depends on the receiver height and the satellite movement. To qualify the observation, the semi-major, semi-minor and the area were determined relative to East London GPS receiver at place at different heights. These different heights of the East London GPS receiver are the initial 1 km, 3 km (resembling a receiver mounted on non pressurised airplane), 10 km (resembling a receiver mounted on a pressurised airplane), 20 km, 100 km and 6500 km (resembling a receiver mounted on a Low Earth Orbit (LEO) satellite), and the results are given by Figure 5.9 to Figure 5.26.

Figure 5.9 to Figure 5.23 show some similar trends and will discuss them in group. The gaps in between (at about 20-65 and 110-155 simulation time steps) show that the satellite was shadowed by the earth and could not see the receiver, and this was expected from the discussion in section 3.5. However, this is not the case for the receiver mounted on LEO satellite which sees the satellite most of the time (Figure 5.24 to Figure 5.26). The Fresnel zone appear to be elliptical in shape throughout the period since the

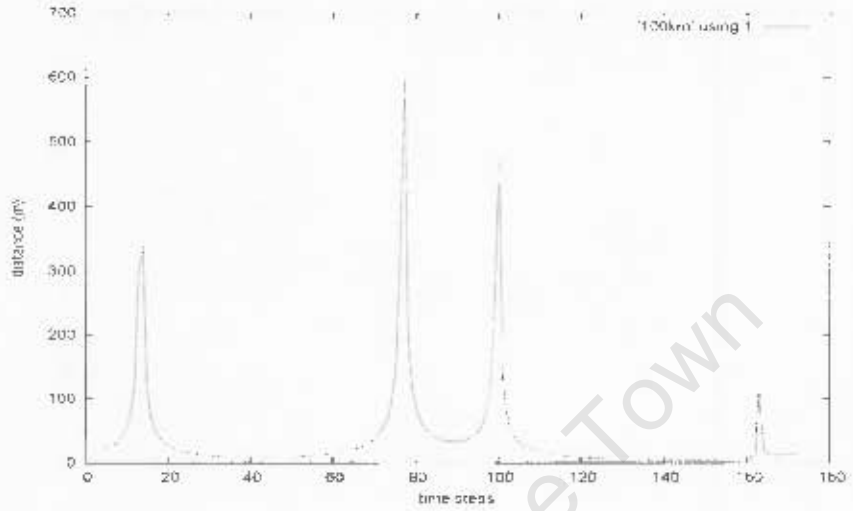

Figure 5.21: Semi-minor of the Fresnel zone (East London receiver placed at 100km height)

semi-major is always greater than the semi-minor and never equals to each other. One can notice the low elevation angles (as discussed in section 3.4) at three different cases, which are resembled by high peak areas of Fresnel zones. These cases are at between 10-20,70-80 and 95-105 simulation time steps and shall be refered to as first,second and third cases respectively. The areas of the ellipses in the first case turn to decrease with an increase in receiver height, and for the second case areas increase with an increase in receiver height, and for the third case the areas decreases with an increase in receiver height. It is important to note that the different cases mentioned above, are relative to satellite positions (see Figure 5.8).

The above discussion is very important for remote sensing application, since it shows the coverage of areas as a function of GPS satellite position (which is the function of time itself) and receiver position. So one does not only have to look at the change in receiver only, but also have to look at the change in GPS satellite position. Having discussed the scenarios for receiver of heights reaching 100km, it is important to notice the change in

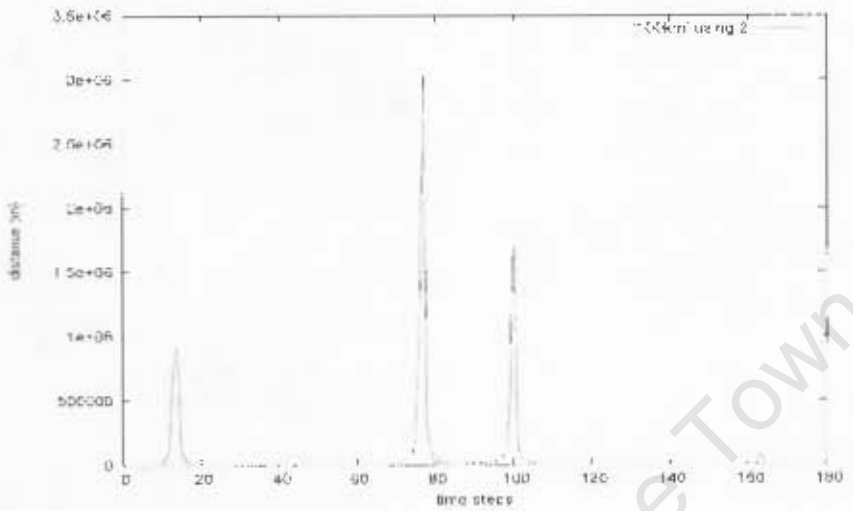

Figure 5.22: Semi-major of the Fresnel zone (East London receiver placed at 100km height)

features(shape) compared to scenario of the receiver placed at the height of 6500 km. The only improvement that this scenario brings is that the GPS satellite sees the receiver most of the time, but it reduces the area of Fresnel zones. This scenario also reduces the afore mentioned three cases into other three different cases (new case on the far right of Figure 5.26). It seems like the elevation angles are generally high for the case of receiver placed on LEO satellite relative to receiver placed at low heights.

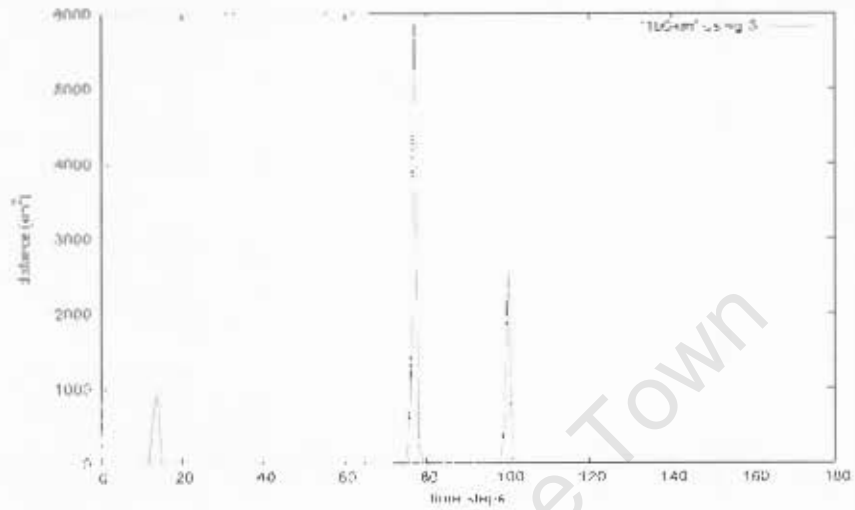

Figure 5.23: Area of the Fresnel zone (East London receiver placed at 100km height)

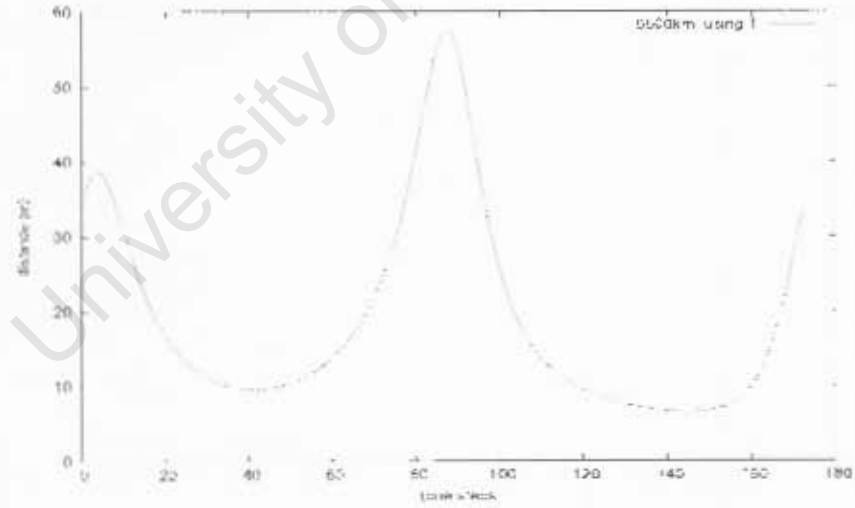

Figure 5.24: Semi-minor of the Fresnel zone (East London receiver placed at 6500km height)

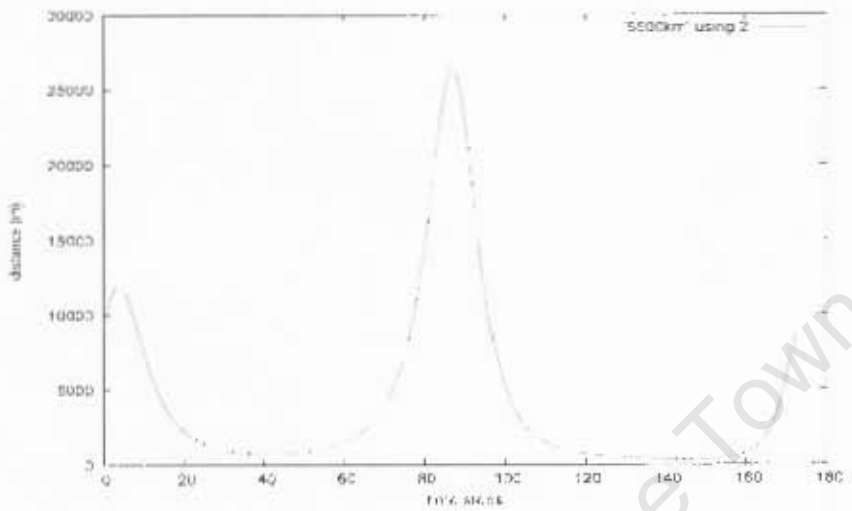

Figure 5.25: Semi-major of the Fresnel zone (East London receiver placed at 6500km height)

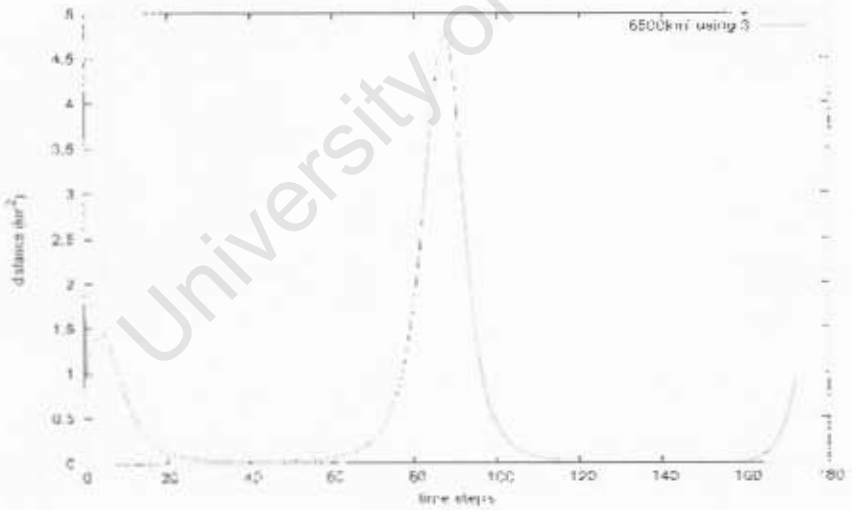

Figure 5.26: Area of the Fresnel zone (East London receiver placed at 6500km height)

Chapter 6

Conclusions and Recommendations for Future Work

The project gives an overview of the GPS system for remote sensing in South Africa. Based on the preliminary observations and results above, the technology of reflected GPS signal for remote sensing is attractive due to the high resolution in space and time, the potentially low cost and the peculiar bistatic, forward scattering geometry which is complementary to the geometry of other techniques. Results of the analysis shows that the GPS receiver placed on East London and Durban are important in analysing the signal since they are located near the reflection point. Although the middle case in between the gaps from the Figure 5.9 - 5.26 the area reaches about 6500 km^2, there are some drawbacks in considering that region for future exploration due to the lower signal strength in the GPS. Therefore the far left case of Figure 5.26 can be considered the region of interest with an area of about 1 km^2.

However the model adopted above is just a begining, it is worth to further persue potential applications for remote sensing. There are couple of sugges-

tion that I think there are important to be considered in the future. First, all satellites passing through South African regions should be considered in future analysis to give better coverage and resolution, not one satellite. In true sense the surface deviates from being an ellipsoid, therefore the estimates of the GPS reflection point would be in error if proper correction to the surface is not made. . If similiar results could be obtained from a receiver on an orbiting satellite, it will provide a unique opportunity to use GPS as a new remote sensing tool on a global scale to infer various geophysical parameters. Soon these reflected GPS signals could be a new source of data for scientist to obtain a better understanding of effects such as global ocean current circulation, global climate change and maybe global warming.

Bibliography

[1] FRENCH G.T.,*Understanding the GPS*, GeoResearch, Inc., Bethesda, page 55-89, 1996.

[2] MASTERS D., ZAVOROTNY V., KATZBERG S.,AND EMERY W.,*GPS Signal Scattering from Land for Moisture Contect Determination*, presented at IGARSS, July 24-28, 2000.

[3] KOMJATHY A., ZAVOROTNY V, AXELRAD P., BORN G., AND GARRISON J.,*GPS signal scattering from sea surface: comparison between experimental data and theoretical model*, presented at the Fifth International Conference on Remote Sensing for Marine and Coastal Environments, San Diego, California, 5-7 October 1998.

[4] MISRA P., AND ENGE P.,*Global Positioning System: Signal, Measurements, and Perfomance*, USA, page 19-46, 2001.

[5] GALILEO,*ESA GPS-like navigation system planned to be operational in 2008*, more information can be found at `http://www.esa.int/navigation/pages/indexGNS.htm`.

[6] LIU J.Y., LIN C.H., TSAI H.F.,AND LIOU Y.A.,*Ionospheric solar flare effects monitored by the ground-based GPS receivers: Theory and observation*, J. Geophys. Res., Vol. 109, 2004.

[7] Phaladi S.G and Cilliers P.J,*Ionospheric changes due to April and July 2000 magnetic storms*, Poster Presentation, 50th South African Institute of Physics (SAIP) annual conference, Pretoria, 2005.

[8] Komjathy, A., Zavorotny, V.U., Axelrad, P., Born, G.H., and Garrison, J.L.,*GPS Signal Scattering from Sea Surface: Wind Speed Retrieval Using Experimental Data and Theoretical Model*, Remote Sensing of Environment, 73:162-174 (2000).

[9] Armatys, M., Masters, D., Komjathy, A., Axelrad, P., and Garrison, J.L., Exploiting GPS as a New Oceanographic Remote Sensing Tool, in the proceedings of the Institute of Navigation Technical Meeting, Anaheim, CA, 26-28 January, 2000.

[10] Zuffada, C.,*Ocean Remote Sensing with GPS*, Journal of Global Positioning System, Vol.1, NO 1:64-65 (2002).

[11] Garrison, J.L., and Katzberg, S.J.,*The Application of Reflected GPS Signals to Ocean Remote Sensing*, Remote Sensing of Environment, 73:175-187 (2000).

[12] Li, Y., Rizos, C., Donskoi, E., Homer, J., and Mojarrabi, B.,*3D Multi-static SAR System for Terrain Imaging Based on Indirect GPS Signals*, Journal of Global Positioning System, Vol.1, NO 1:34-39 (2002).

[13] Chang, H.C., *Assessment of multi-source remote sensing DEMs with RTK-GPS and levelling surveys*, 18th Int. Tech. Meeting of the Satellite Division of the U.S. Institute of Navigation, Long Beach, California, 13-16 September 2005.

[14] Masters, D., P. Axelrad, V. Zavorotny, S. J. Katzberg, F. Lalezari, *A Passive GPS Bistatic Radar Altimeter for Aircraft Navigation*, Proceedings of the ION-GPS, Salt Lake City, 2001.

[15] Lowe, S.T., C. Zuffada, Y. Chao, P. Kroger, J.L. LaBreque and L.E. Young, *5-cm precision aircraft ocean altimetry using GPS reflections*, Geophys. Res. Lett., Vol. 29, NO.10, 10.1029/2002GL014759, 2002.

[16] Komjathy, A., J.A. Maslanik, V.U. Zavorotny, P. Axelrad, and S.J. Katzberg,*Towards GPS Surface Reflection Remote Sensing of Sea Ice Conditions*, In the Proceedings of the Sixth International Conference on Remote Sensing for Marine and Coastal Environments, Charleston, SC, 1-3 May, Vol II, pp. 447-456, 2000.

[17] Gibbons, G.,*GPS World's Big Book of GPS*, Advanstar Communications, USA, page 17-98, 1999.

[18] Wells, D.E., N. Beck, D. Delikaraoglou, A. Kleusberg., E.J. Krakiwsky, G. Lachapelle, R.B. Langley, M. Nakiboglu, K.P. Schwarz, J.M. Tranquilla, and P. Vancek , Guide to GPS Positioning, Canadian GPS Associates, Fredericton, N.B., Canada, 1987.

[19] Garrison, J.L., Komjathy, A., V.U. Zavorotny, and S.J. Katzberg, *Wind Speed Measurement Using Forward Scattered GPS Signals*, IEEE Transactions on GeoScience and Remote Sensing, Vol. 40, NO. 1, January 2002.

[20] Stolk, K.,*Bistatic Sensing with Reflected GPS Signals Observed With a Digital Beam-Steered Antenna Array*, Proceedings of ION GPS 2003, Portland, Oregon, September 2003.

[21] Komjathy, A., V.U. Zavorotny, P. Axelrad, G. Born, and J. Garrison,*GPS Signal Scattering from Sea Surface: Comparison Between Experimental Data and Theoretical Model*, Proceedings at the Fifth International Conference on Remote Sensing for Marine and Coastal Environments, San Diego, California, 5-7 October 1998.

[22] Zavorotny V.U., P. Axelrad, G. Born, and J. Garrison,*Scattering of GPS signals from the ocean with wind remote sensing applications*. IEEE Transactions in Geoscience and Remote Sensing, Vol. 38, No. 2, pp. 951-964, 2000.

[23] Komjathy A., Garrison J.,and Zavorotny V.U.,*GPS: A New Tool for Ocean Science* , GPS World, pp. 50-56, April 1999.

[24] Komjathy A., M. Armatys, D. Masters, P. Axelra, V.U. Zavorotny, S.J. Katzberg,*Development in Using GPS for Oceanographic Remote Sensing: Retrieval of Ocean Surface Wind Speed and Wind Direction*, Presented at the ION 2001 National Technical Meeting, Long Beach, CA, 22-24 January 2001.

[25] Ulaby F.T., Moore R.K., and Fung A.K.,*Microwave Remote Sensing Active and Passive: Radar Remote Sensing and Surface Scattering and Emission Theory*, Vol. 2, Canada, pp. 825-828, 1982.

[26] Cardellach E., Trenhaft R., Franklin G., Gorelik J., Lowe S., and Young L.,*Carrier Phase Delay Altimetry from Low Elevation GPSR Measurements*, NASA/JPL Jet Propulsion Laboratory, USA, 2002.http://starlab.es/gnssr2003/Proceedings.html

[27] Ray J.K.,*Mitigation of GPS Code and Carrier Phase Multipath Effects Using a Multi-Antenna System*. PhD Thesis, University of Calgary, Canada, 2000.

[28] Beckmann P., and Spizzichino A., *The Scattering of Electromagnetic Waves from Rough Surfaces*, Pergamon Press, Oxford, pp. 9-15, 1963.

[29] Klukas R., Julien O., Dong L., Cannon E., and Lachapelle G., *Effects of building materials on UHF ranging signals*, GPS Solutions, Vol. 8, pp. 1-8, 2004.

[30] KOMJATHY A.,*Application of GPS As a New Remote Sensing Tool: Retrieval of Ocean Surface Wind Speed*. Presented at the University of New Brunswick, Fredericton, NB, Canada, 27 October 1998.

[31] MORTENSEN M.D., AND HEG P., *Inversion of GPS occultation measurements using Fresnel diffraction theory*, GEOPHYSICAL RESEARCH LETTERS, VOL. 25, NO. 13, PAGES 2441-2444, JULY 1, 1998.

[32] ERICA BRICCHI, JOHN D. MILLS, PETER G. KAZANSKY, AND BRUCE G. KLAPPAUF, *Birefringent Fresnel zone plates in silica fabricated by femtosecond laser machining*, OPTICS LETTERS, Vol. 27, No. 24, December 15, 2002.

[33] LIANG Y.,*MULTIPATH FRESNEL ZONE ROUTING FOR WIRELESS AD HOC NETWORKS*, MSc Thesis, State University, Blacksburg, Virginia, 2004.

[34] SOLER T., *A compendium of transformation formulas useful in GPS work*, Journal of Geodesy, Vol. 72, pp. 482-490, 1998.

[35] SEEBER G., *Satellite Geodesy*, 2nd Edition, de Guyter, New York, pp. 2-30, 2003.

[36] MONTENBRUCK O. AND GILL E.,*Satellite Orbits: Models, Methods, and Applications*, Springer, Berlin, Heidelberg, New York, 2000.

[37] S. WU, T. MEEHAM AND L. YOUNG,*The Potential Use of GPS Signals as Ocean Altimetry Observable*, National Technical Meeting, Santa Monica, CA, 14 - 16 January 1997.

[38] S.J. KARTZBERG AND J.L. GARRISON,*Utilizing GPS To Determine Inospheric Delay Over the Ocean*, NASA Technical Memorandum 4750, 1996.

Appendix

I have written the following IDL procedures entirely on my own:

- "reflection_pos.pro"
- "view_gps_system.pro"
- "fresnel.pro"
- "fresnel_ellipse.pro"

And the following procedures have been modified from the original ones authored by Dr. R. Lord:

- "compile_routines.pro"
- "convert_latlon_xyz.pro"
- "convert_xyz_latlon.pro"
- "define_gps_params.pro"
- "get_gps_orbit_parameters.pro"
- "plot_map.pro"
- "satellite_inertial_vectors.pro"

And the rest are being sorely written by Dr. R. Lord.

```
;=============================================================================================

 Description:
 ------------
 Procedure to compile all routines and functions.

 Code maintainer:
 ---------------------------------------------------------------------------------------------
 Shikoane Given Phaladi, 02-06-06

;=============================================================================================

pro compile_routines

resolve_routine, 'convert_xyz_latlon'
resolve_routine, 'crossp', /is_function
resolve_routine, 'define_gps_params'
resolve_routine, 'ellipsoidal_intersection'
resolve_routine, 'get_repeat_cycle_semimajor'
resolve_routine, 'get_gps_orbit_parameters'
resolve_routine, 'filepath', /is_function
resolve_routine, 'map_continents'
resolve_routine, 'map_grid'
resolve_routine, 'map_set'
resolve_routine, 'norm', /is_function
resolve_routine, 'path_sep', /is_function
resolve_routine, 'plot_map'
resolve_routine, 'reverse', /is_function
resolve_routine, 'rotx', /is_function
resolve_routine, 'roty', /is_function
resolve_routine, 'rotz', /is_function
resolve_routine, 'satellite_inertial_vectors'
resolve_routine, 'transform_inertial_to_fixed'
resolve_routine, 'reflection_pos'
resolve_routine, 'fresnel'
resolve_routine, 'reflection_angle'
resolve_routine, 'fresnel_smajor'
resolve_routine, 'fresnel_ellipse'

end
```

```
COMMON gps, constants, instrument_input_parameters, instrument_calculated_parameters, orbit_param
eters, scene_parameters , scene_calculated_parameters, target_positions, program_execution
```

```
;===================================================================
;
; Description:
; ------------
; Procedure to obtain xyz position of poing on Earth's surface
; given geodetic latitude and longitude according to WSG84.
;
; References:
; -----------
; Misra P, and Enge P.: "Global Positioning System: Signal,

;                        Measurements, and Perfomance", p.115.
;
; T. Soler: "A compedium of transformation formulas useful in GPS work",
;           J. Geodesy,17:482-490, 1998.
;
; Input:
; ------
; latlonh_pos                  - vector or array of vectors containing
;                                latitude, longitude, height in [rad, rad, m]
;
; Output:
; -------
; xyz_pos                      - vector or array of vectors containing
;                                earth fixed xyz position in [m,m,m]
; Original Code maintainer:
; -------------------------
; Elke Boerner, 24 April 2002
;
; Modified and updated by:
; ------------------------
; Shikoane Given Phaladi, 03 May 2006
;
;===================================================================

Pro convert_latlon_xyz, latlonh_pos, xyz_pos=xyz_pos


;----- Definition of global variables
@introduce_common_parameters.bat


;-----Get parameters
semi_major            = constants.semi_major
flattening            = constants.flattening


;-----Calculate eccentricity squared
ecc2                  = 2d*flattening - flattening^2d


;-----Find out if xyz_pos is a vector, or an array of vectors.
size_pos              = size(latlonh_pos, /N_DIMENSIONS)

;-----Calculate latitude, longitude and height according to WGS84
if (size_pos eq 2) then begin
   numpoints          = n_elements(latlonh_pos[*, 0])
   xyz_pos            = dblarr(numpoints, 3)

   for i=0L, (numpoints-1) do begin
        n_quer        = semi_major / sqrt(1d - (ecc2 * (sin(latlonh_pos[i,0]))^2d) )
        xvalue        = (n_quer + latlonh_pos[i,2])*cos(latlonh_pos[i,0])*cos(latlonh_pos[i,1])
        yvalue        = (n_quer + latlonh_pos[i,2])*cos(latlonh_pos[i,0])*sin(latlonh_pos[i,1])
        zvalue        = ((n_quer*(1d - ecc2) + latlonh_pos[i,2])*sin(latlonh_pos[i,0]))

        xyz_pos[i,*]  = [xvalue, yvalue, zvalue]
   endfor

endif else begin
        n_quer        = semi_major / sqrt(1d - (ecc2 * (sin(latlonh_pos[0]))^2d) )
        xvalue        = (n_quer + latlonh_pos[2])*cos(latlonh_pos[0])*cos(latlonh_pos[1])
        yvalue        = (n_quer + latlonh_pos[2])*cos(latlonh_pos[0])*sin(latlonh_pos[1])
```

```
        zvalue          = ((n_quer*(1d - ecc2) + latlonh_pos[2])*sin(latlonh_pos[0]))

        xyz_pos         = [xvalue, yvalue, zvalue]
endelse

End
```

```
;=============================================================================
;
; Description:
; ------------
; Procedure to obtain geodetic latitude [rad], longitude [rad] and  height[m]
; coordinates from cartesian coordinates (WGS84).
;
; References:
; -----------
; G. Seeber: "Satellite geodesy: foundations, methods, and applications", p.20
; Misra P., and Enge P.: "Global Positioning System: Signal, Measurements, and
;                         perfomance", USA, pg 115, 2004
; Input:
; ------
; xyz_pos                   - vector or array of vectors containing
;                             earth fixed xyz position in [m]
;
; Output:
; -------
; latlonh_pos               - vector or array of vectors containing
;                             latitude,longitude,height in [rad,rad,m]
;
; Code maintainer:
; ----------------
; Richard Lord, 09-07-2004
; Modified by:
; ------------
; Shikoane Phaladi, 03 May 2004
;=============================================================================


Pro convert_xyz_latlon, xyz_pos, latlonh_pos=latlonh_pos


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Get parameters
semi_major            = constants.semi_major
flattening            = constants.flattening


;-----Calculate eccentricity squared
ecc2                  = 2d*flattening - flattening^2d


;-----Set start values for iterative calculation
accuracy              = 1.0d-15
error_phi             = 1d
error_height          = 1d
n_quer                = 0.001d
height                = 0.01d
height_neu            = 0.0001d
phi                   = 0.01d
phi_neu               = 0.0001d


;-----Find out if xyz_pos is a vector, or an array of vectors.
size_pos              = size(xyz_pos, /N_DIMENSIONS)


;-----Calculate latitude, longitude and height according to WGS84 (iterative calculation)
if (size_pos eq 2) then begin
   numpoints          = n_elements(xyz_pos[*, 0])
   latlonh_pos        = dblarr(numpoints, 3)

   for i=0L, (numpoints-1) do begin
      ; start values for each target!!
      error_phi       = 1d
      error_height    = 1d
      n_quer          = 0.001d
```

```
      height            = 0.01d
      height_neu        = 0.0001d
      phi               = 0.01d
      phi_neu           = 0.0001d

      ; calculation of the longitude
      longitude         = atan(xyz_pos[i, 1], xyz_pos[i, 0])

      ; calculation of phi (latitude) and height for the given accuracy in an iterative method
      while ((accuracy lt abs(error_phi)) AND (accuracy lt abs(error_height))) do begin
         height         = height_neu
         phi            = phi_neu
         n_quer         = semi_major / sqrt(1d - (ecc2 * (sin(phi_neu))^2d) )
         p_quer         = sqrt((xyz_pos[i, 0])^2d + (xyz_pos[i, 1])^2d)
         height_neu     = ( p_quer / cos(phi) ) - n_quer
         phi_neu        = atan(    (xyz_pos[i, 2] / p_quer) * $
                          1d / (1d - ecc2 * n_quer / (n_quer + height)) )
         error_phi      = phi_neu - phi
         error_height   = height_neu - height
      endwhile

      ; define output array
      latlonh_pos[i, *] = [phi_neu, longitude, height_neu]
   endfor

endif else begin
   ; calculation of the longitude
   longitude           = atan(xyz_pos[1], xyz_pos[0])

   ; calculation of phi (latitude) and height for the given accuracy in an iterative method
   while ((accuracy lt abs(error_phi)) AND (accuracy lt abs(error_height))) do begin
      height           = height_neu
      phi              = phi_neu
      n_quer         = semi_major / sqrt(1d - (ecc2 * (sin(phi_neu))^2d) )
         p_quer         = sqrt((xyz_pos[0])^2d + (xyz_pos[1])^2d)
         height_neu     = ( p_quer / cos(phi) ) - n_quer
         phi_neu        = atan(    (xyz_pos[2] / p_quer) * $
                          1d / (1d - ecc2 * n_quer / (n_quer + height)) )
      error_phi        = phi_neu - phi
      error_height     = height_neu - height
   endwhile

   ; define output vector for one target
   latlonh_pos         = [phi_neu, longitude, height_neu]

endelse

End
```

```
;=============================================================================
;
; Description:
; ------------
; Procedure to set global GPS satellite orbit parameters, by reading the gps almanac
; files from the website "hppt://www.navcen.uscg.gov/ftp/GPS/almanacs/"
;
;
; Outputs:
; --------
; orbit parameters [global]: sat_name     []     : satellite name
;                            eccentricity []     : orbit eccentricity
;                            inclination  [rad]  : inclination
;                            semi_major   [m]    : semi major axis of orbit
;                            omega        [rad]  : angle of perigee
;                            asc_node     [rad]  : right ascencion of ascending node
;                                  mean_anomaly [rad] : mean anomaly
;
; Code maintainer:
; ----------------
; Richard Lord, 08-07-2004
;
; Updated by:
; -----------
; Shikoane Given Phaladi, 04 May 2006
;
;=============================================================================


Pro get_gps_orbit_parameters

;-----Definition of global variables
@introduce_common_parameters.bat

;-----Get global parameters
filename    = program_execution.gps_orbit_data
num_gps_ids = n_elements(orbit_parameters.sat_name)

;-----Initialise
sat_name     = strarr(num_gps_ids)
eccentricity = dblarr(num_gps_ids)
inclination  = dblarr(num_gps_ids)
semi_major   = dblarr(num_gps_ids)
omega        = dblarr(num_gps_ids)
asc_node     = dblarr(num_gps_ids)
mean_anomaly = dblarr(num_gps_ids)

;-----Read file
line    = ''
counter = 0L
openr, lun, filename, /get_lun, ERROR=err
if (err eq 0) then begin
   while (not eof(lun)) do begin
      readf, lun, line
      if (strlowcase(strmid(strtrim(line, 2), 0, 3)) eq 'id:') then begin
         sat_name[counter] = strtrim(strmid(line, 3), 2)
         for i=0L, 10 do begin
            readf, lun, line
            label = strlowcase(strtrim(strmid(line, 0, strpos(line, ':')), 2))
            value = strmid(strmid(line, strpos(line, ':') + 1), 2)
            if (label eq 'eccentricity') then eccentricity[counter] = double(value)
            if (label eq 'orbital inclination(rad)') then inclination[counter] = double(value)
            if (label eq 'sqrt(a)  (m 1/2)') then semi_major[counter] = double(value) ^ 2d
            if (label eq 'right ascen at week(rad)') then asc_node[counter] = double(value)
            if (label eq 'argument of perigee(rad)') then omega[counter] = double(value)
            if (label eq 'mean anom(rad)') then mean_anomaly[counter] = double(value)
         endfor
         counter = counter + 1L
      endif
   endwhile
   free_lun, lun
endif else begin
```

```
   message, 'Error: file ' + filename + ' not found!'
endelse

;----- Update global parameters
orbit_parameters.sat_name      = sat_name
orbit_parameters.eccentricity = eccentricity
orbit_parameters.inclination  = inclination
orbit_parameters.semi_major    = semi_major
orbit_parameters.omega         = omega
orbit_parameters.asc_node      = asc_node
orbit_parameters.mean_anomaly = mean_anomaly

;----- Print information
if (program_execution.verbose_mode eq 1) then begin
   print, strjoin(replicate('-', program_execution.dotted_line))
   print, "Global variables updated by 'set_satellite_orbit_parameters' procedure:"
   print, strjoin(replicate('-', program_execution.dotted_line))
endif

;----- Adjust semi-major axis to get an exact 1 day repeat cycle
if (program_execution.adjust_semimajor eq 1) then begin
   for i=0L, (num_gps_ids - 1L) do begin
      if (orbit_parameters.sat_name[i] ne '') then get_repeat_cycle_semimajor, i
   endfor
endif

End
```

```
;=============================================================================
;
; Description:
; ------------
; Procedure to initialise global simulation parameters.
;
; Reference:
; ----------
; The constants have been take from the table in pg 81 of (Misra P. and Enge P.,
; "Global Position System: Signal, Measurements, and Perfomance", USA, 2004)
;
; Code maintainer:
; ----------------
; Richard Lord
;
; Updated by:
; -----------
; Shikoane Phaladi
; 03 May 2006
;
;=============================================================================

Pro define_gps_params


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Initialise
radeg       = 180d / !dpi
dtor        = !dpi/180d
num_gps_ids = 31L


;-----------------------------------------------------------------------------------------
constants={                                          $
      radeg             : radeg                 ,$   ;exact factor of conversion rad and deg
      dtor                   : dtor                  ,$   ;exact factor to convert deg to rad
      co                : 2.99792458d8          ,$   ;speed of light [m/s] (exact)
      gm_earth          : 3986004.415d8         ,$   ;G * mass_earth   [m^3/s^2]
      semi_major             : 6378137.0d            ,$   ;WGS 84, revised in 1997 [m]
      flattening             : 3.457131d-3           ,$   ;flattening
      semi_minor        : 6356.75231425d3       ,$   ;WGS 84, revised in 1997 [m]
      omega_earth       : 7292115.0d-11          $   ;angular velocity [1/s] WGS84
          }
;-----------------------------------------------------------------------------------------

;-----------------------------------------------------------------------------------------
orbit_parameters={                                   $
      sat_name                : strarr(num_gps_ids) ,$   ;GPS ID
      eccentricity            : dblarr(num_gps_ids) ,$   ;eccentricity
      inclination             : dblarr(num_gps_ids) ,$   ;inclination [rad]
      semi_major              : dblarr(num_gps_ids) ,$   ;semi major axis of orbit [m]
      omega                   : dblarr(num_gps_ids) ,$   ;argument of perigee [rad]
      asc_node                : dblarr(num_gps_ids) ,$   ;right ascencion of ascending node [rad]
      mean_anomaly            : dblarr(num_gps_ids)  $   ;mean anomaly []
                }
;-----------------------------------------------------------------------------------------


;-----------------------------------------------------------------------------------------
instrument_input_parameters={                        $
      leftlook                : 0                   ,$   ;set to 1 if antenna is left-looking
      yaw                     : 0d                  ,$   ;yaw angle    [rad]
      roll                    : 0d                  ,$   ;roll angle   [rad]
      pitch                   : 0d                  ,$   ;pitch angle  [rad]
      squint                  : 0d                   $   ;squint angle [rad]
          }
;-----------------------------------------------------------------------------------------
```

```
;-------------------------------------------------------------------------------------------
program_execution={                                    $
      dotted_line             : 71                    ,$  ;length of dotted line
      gps_orbit_data          : 'gps_almanac.txt'     ,$  ;file containing GPS orbit data
      verbose_mode            : 1                     ,$  ;print details, 0=no, 1=yes
      adjust_semimajor        : 1                     ,$  ;adjust semi-major axis, 0=no, 1=yes
      repeat_cycle_days       : 1                     ,$  ;period for repeat cycle
      nadir_definition        : 'perpendicular'        $  ;through 'center' or 'perpendicular' to E
arth
                  }
;-------------------------------------------------------------------------------------------


;-------------------------------------------------------------------------------------------
if (program_execution.verbose_mode eq 1) then begin
   print, ' '
   print, strjoin(replicate('-', program_execution.dotted_line))
   print, strjoin(replicate('-', program_execution.dotted_line))
   print, "Global variables have been initialised by 'define_gps_params' procedure."
   print, strjoin(replicate('-', program_execution.dotted_line))
endif
;-------------------------------------------------------------------------------------------

End
```

```
;=============================================================================
;
; Description:
; ------------
; Procedure to calculate the intersection of the antenna pointing vector with
; the Earth ellipsoid.
;
; Inputs:
; -------
; sat_pos_in    - (x,y,z) satellite position vector in inertial system
; sat_vel_in    - (x,y,z) satellite velocity vector in inertial system
; off_nadir     - off-nadir angle [rad]
; squint        - squint angle [rad]
;
; Outputs:
; --------
; target_pos_in - target (x,y,z) position on Earth ellipsoid in inertial system
; range         - range from satellite position to target position
;
; Code maintainer:
; ----------------
; Richard Lord, 25-04-2002
; copyright DLR
;
; References:
; -----------
; X-SAR Geometric Error Budget Analysis, Univ. of Zurich, Nov. 1990, pg. 24
; R.T. Lord and E. Boerner, 'Calculation of Earth Ellipsoid Intersection',
; DLR, July 2002.
;
;=============================================================================


Pro ellipsoidal_intersection, sat_pos_in, sat_vel_in, off_nadir, squint=squint, $
                              target_pos_in=target_pos_in, range=range


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Default values
if (not n_elements(squint)) then squint=0d


;-----Get constants and antenna parameters
nadir_definition = program_execution.nadir_definition
r_equatorial     = constants.semi_major
r_polar          = constants.semi_minor
leftlook         = instrument_input_parameters.leftlook
yaw              = instrument_input_parameters.yaw
roll             = instrument_input_parameters.roll
pitch            = instrument_input_parameters.pitch


;-----Set antenna look direction
if keyword_set(leftlook) then begin
   antenna_direction = -1           ;-1 for left-looking antenna
endif else begin
   antenna_direction = 1            ;+1 for right-looking antenna
endelse


;-----Error checking
Size_pos = size(sat_pos_in)
Size_vel = size(sat_vel_in)
if (Size_pos[0] ne Size_vel[0]) then begin
   message, 'Error: Position and velocity vectors have different sizes!'
endif


;-----Convert xyz to latlonh
```

```
convert_xyz_latlon, sat_pos_in, latlonh=sat_latlonh


;-----Find out if sat_pos and sat_vel are vectors, or arrays of vectors.
if (Size_pos[0] eq 2) then numvectors = Size_pos[1] else numvectors = 1

if (numvectors eq 1) then begin
   sat_pos_in  = reform(sat_pos_in, 1, 3)
   sat_vel_in  = reform(sat_vel_in, 1, 3)
   sat_latlonh = reform(sat_latlonh, 1, 3)
endif

range         = dblarr(numvectors)
target_pos_in = dblarr(numvectors, 3)


n_satellite_system = [cos(off_nadir - (!dpi/2d)) * cos(squint) * antenna_direction, $
                      cos(off_nadir - (!dpi/2d)) * sin(squint), $
                      sin(off_nadir - (!dpi/2d))]

n_trajectory_system = reform(rotz(yaw) ## roty(roll) ## rotx(pitch) ## n_satellite_system)


for i=0L, (numvectors-1) do begin


   ;-----Find the trajectory system directional unit vectors
   if (nadir_definition eq 'perpendicular') then begin
      zenith = [cos(sat_latlonh[i, 0]) * cos(sat_latlonh[i, 1]), cos(sat_latlonh[i, 0]) * $
                sin(sat_latlonh[i, 1]), sin(sat_latlonh[i, 0])]
   endif else begin
      ;Note: In the following definition of zenith the vector goes through the origin!
      zenith = reform(sat_pos_in[i, *])
   endelse

   xdir = crossp(sat_vel_in[i, *], zenith)
   ydir = crossp(zenith, xdir)
   ez   = zenith / norm(zenith)
   ex   = xdir   / norm(xdir)
   ey   = ydir   / norm(ydir)

   D = [transpose(ex), transpose(ey), transpose(ez)]

   n_inertial_system = reform(D ## n_trajectory_system)


   ;-----A line can be described through a point P=(x1, y1, z1) and a direction
   ;vector r=(a, b, c).  The parametric representation for the line is:
   ;x = x1 + a*lambda
   ;y = y1 + b*lambda
   ;z = z1 + c*lambda


   ;-----The equation for the earth ellipsoid is:
   ;(x^2 / semimajor^2) + (y^2 / semimajor^2) + (z^2 / semiminor^2) - 1 = 0


   ;-----Now insert the equations for the line into the ellipsoid equation and solve
   ;for lambda.

   a = ((n_inertial_system[0]^2d + n_inertial_system[1]^2d) / (r_equatorial^2d)) + $
       ((n_inertial_system[2]^2d) / r_polar^2d)
   b = (((2d * n_inertial_system[0] * sat_pos_in[i, 0]) + (2d * n_inertial_system[1] * $
       sat_pos_in[i, 1])) / (r_equatorial^2d)) + ((2d * n_inertial_system[2] * $
       sat_pos_in[i, 2]) / (r_polar^2d))
   c = (((sat_pos_in[i, 0]^2d) + (sat_pos_in[i, 1]^2d)) / (r_equatorial^2d)) + $
       ((sat_pos_in[i, 2]^2d) / (r_polar^2d)) - 1d


   sqrt_arg = b^2d - 4*a*c
   if (sqrt_arg ge 0d) then begin
      lambda1 = (-b + sqrt(sqrt_arg)) / (2d * a)
```

```
      lambda2 = (-b - sqrt(sqrt_arg)) / (2d * a)

      ;-----We are only interested in the smaller value of lambda, i.e. we don't
      ;want the point where the line exits the ellipsoid.

      lambda = min([lambda1, lambda2])
      range[i] = lambda

      target_pos_in[i, *] = sat_pos_in[i, *] + (n_inertial_system * lambda)

   endif else begin
      print, 'Warning: No real solution for earth trace!'
      target_pos_in[i, *] = [1, 1, 1]  ;To prevent division by 0 when converting to lat/lon
      range[i]            = -1         ;i.e. an impossible value
   endelse

endfor


if (numvectors eq 1) then begin
   sat_pos_in    = reform(sat_pos_in)
   sat_vel_in    = reform(sat_vel_in)
   target_pos_in = reform(target_pos_in)
   range         = range[0]
endif


End
```

```
;=============================================================================================================
;
; Description:
; ------------
; Procedure to compile all routines and functions.
;
; Code maintainer:
; ----------------
; Shikoane Given Phaladi, 02-06-06
;
;
;=============================================================================================================


Pro compile_routines

resolve_routine, 'convert_xyz_latlon'
resolve_routine, 'crossp', /is_function
resolve_routine, 'define_gps_params'
resolve_routine, 'ellipsoidal_intersection'
resolve_routine, 'get_repeat_cycle_semimajor'
resolve_routine, 'get_gps_orbit_parameters'
resolve_routine, 'filepath', /is_function
resolve_routine, 'map_continents'
resolve_routine, 'map_grid'
resolve_routine, 'map_set'
resolve_routine, 'norm', /is_function
resolve_routine, 'path_sep', /is_function
resolve_routine, 'plot_map'
resolve_routine, 'reverse', /is_function
resolve_routine, 'rotx', /is_function
resolve_routine, 'roty', /is_function
resolve_routine, 'rotz', /is_function
resolve_routine, 'satellite_inertial_vectors'
resolve_routine, 'transform_inertial_to_fixed'
resolve_routine, 'reflection_pos'
resolve_routine, 'fresnel'
resolve_routine, 'reflection_angle'
resolve_routine, 'fresnel_smajor'



resolve_routine, 'fresnel_ellipse'
resolve_routine, 'tester'


End
```

```
;=============================================================================
;                                                                            |
; Description:                                                               |
; ------------                                                               |
; Procedure to obtain xyz position of poing on Earth's surface               |
; given geodetic latitude and longitude according to WSG84.                  |
;                                                                            |
; References:                                                                |
; -----------                                                                |
; Misra P, and Enge P.: "Global Positioning System: Signal,                  |
;                        Measurements, and Perfomance", p.115.               |
;                                                                            |
; T. Soler: "A compedium of transformation formulas useful in GPS work",     |
;            J. Geodesy,17:482-490, 1998.                                    |
;                                                                            |
; Input:                                                                     |
; ------                                                                     |
; latlonh_pos                   - vector or array of vectors containing      |
;                                 latitude, longitude, height in [rad, rad, m] |
;                                                                            |
; Output:                                                                    |
; -------                                                                    |
; xyz_pos                       - vector or array of vectors containing      |
;                                 earth fixed xyz position in [m,m,m]        |
; Original Code maintainer:                                                  |
; -------------------------                                                  |
; Elke Boerner, 24 April 2002                                                |
;                                                                            |
; Modified and updated by:                                                   |
; ------------------------                                                   |
; Shikoane Given Phaladi, 03 May 2006                                        |
;                                                                            |
;=============================================================================

Pro convert_latlon_xyz, latlonh_pos, xyz_pos=xyz_pos


;----- Definition of global variables
@introduce_common_parameters.bat


;-----Get parameters
semi_major            = constants.semi_major
flattening            = constants.flattening


;-----Calculate eccentricity squared
ecc2                  = 2d*flattening - flattening^2d


;-----Find out if xyz_pos is a vector, or an array of vectors.
size_pos              = size(latlonh_pos, /N_DIMENSIONS)

;-----Calculate latitude, longitude and height according to WGS84 (iterative
calculation)
if (size_pos eq 2) then begin
   numpoints          = n_elements(latlonh_pos[*, 0])
   xyz_pos            = dblarr(numpoints, 3)

   for i=0L, (numpoints-1) do begin
        n_quer        = semi_major / sqrt(1d - (ecc2 * (sin(latlonh_pos[i,0]))^2d) )
```

```
        xvalue          = (n_quer + latlonh_pos[i,2])*cos(latlonh_pos[i,0])*cos
(latlonh_pos[i,1])
        yvalue          = (n_quer + latlonh_pos[i,2])*cos(latlonh_pos[i,0])*sin
(latlonh_pos[i,1])
        zvalue          = ((n_quer*(1d - ecc2) + latlonh_pos[i,2])*sin(latlonh_pos[i,0]))

        xyz_pos[i,*]    = [xvalue, yvalue, zvalue]
   endfor

endif else begin
        n_quer          = semi_major / sqrt(1d - (ecc2 * (sin(latlonh_pos[0]))^2d) )
        xvalue          = (n_quer + latlonh_pos[2])*cos(latlonh_pos[0])*cos(latlonh_pos
[1])
        yvalue          = (n_quer + latlonh_pos[2])*cos(latlonh_pos[0])*sin(latlonh_pos
[1])
        zvalue          = ((n_quer*(1d - ecc2) + latlonh_pos[2])*sin(latlonh_pos[0]))

        xyz_pos         = [xvalue, yvalue, zvalue]
endelse

End
```

```
;=============================================================================================
;
; Description:
; ------------
; Procedure to obtain geodetic latitude [rad], longitude [rad] and  height[m]
coordinates
; from cartesian coordinates (WGS84).
;
; References:
; -----------
; G. Seeber: "Satellite geodesy: foundations, methods, and applications", p.20
; Misra P., and Enge P.: "Global Positioning System: Signal, Measurements, and
;                         perfomance", USA, pg 115, 2004
; Input:
; ------
; xyz_pos                   - vector or array of vectors containing
;                             earth fixed xyz position in [m]
;
; Output:
; -------
; latlonh_pos               - vector or array of vectors containing
;                             latitude,longitude,height in [rad,rad,m]
;
; Code maintainer:
; ----------------
; Richard Lord, 09-07-2004
; Modified by:
; ------------
; Shikoane Phaladi, 03 May 2004
;=============================================================================================


Pro convert_xyz_latlon, xyz_pos, latlonh_pos=latlonh_pos


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Get parameters
semi_major          = constants.semi_major
flattening          = constants.flattening


;-----Calculate eccentricity squared
ecc2                = 2d*flattening - flattening^2d


;-----Set start values for iterative calculation
accuracy            = 1.0d-15
error_phi           = 1d
error_height        = 1d
n_quer              = 0.001d
height              = 0.01d
height_neu          = 0.0001d
phi                 = 0.01d
phi_neu             = 0.0001d


;-----Find out if xyz_pos is a vector, or an array of vectors.
size_pos            = size(xyz_pos, /N_DIMENSIONS)
```

```
;-----Calculate latitude, longitude and height according to WGS84 (iterative
calculation)
if (size_pos eq 2) then begin
   numpoints           = n_elements(xyz_pos[*, 0])
   latlonh_pos         = dblarr(numpoints, 3)

   for i=0L, (numpoints-1) do begin
      ; start values for each target!!
      error_phi        = 1d
      error_height     = 1d
      n_quer           = 0.001d
      height           = 0.01d
      height_neu       = 0.0001d
      phi              = 0.01d
      phi_neu          = 0.0001d

      ; calculation of the longitude
      longitude        = atan(xyz_pos[i, 1], xyz_pos[i, 0])

      ; calculation of phi (latitude) and height for the given accuracy in an iterative
method
      while ((accuracy lt abs(error_phi)) AND (accuracy lt abs(error_height))) do begin
         height        = height_neu
         phi           = phi_neu
         n_quer        = semi_major / sqrt(1d - (ecc2 * (sin(phi_neu))^2d) )
         p_quer        = sqrt((xyz_pos[i, 0])^2d + (xyz_pos[i, 1])^2d)
         height_neu    = ( p_quer / cos(phi) ) - n_quer
         phi_neu       = atan(   (xyz_pos[i, 2] / p_quer) * $
                         1d / (1d - ecc2 * n_quer / (n_quer + height)) )
         error_phi     = phi_neu - phi
         error_height  = height_neu - height
      endwhile

      ; define output array
      latlonh_pos[i, *] = [phi_neu, longitude, height_neu]
   endfor

endif else begin
   ; calculation of the longitude
   longitude           = atan(xyz_pos[1], xyz_pos[0])

   ; calculation of phi (latitude) and height for the given accuracy in an iterative
method
   while ((accuracy lt abs(error_phi)) AND (accuracy lt abs(error_height))) do begin
      height           = height_neu
      phi              = phi_neu
      n_quer          = semi_major / sqrt(1d - (ecc2 * (sin(phi_neu))^2d) )
         p_quer         = sqrt((xyz_pos[0])^2d + (xyz_pos[1])^2d)
         height_neu     = ( p_quer / cos(phi) ) - n_quer
         phi_neu        = atan(   (xyz_pos[2] / p_quer) * $
                          1d / (1d - ecc2 * n_quer / (n_quer + height)) )
      error_phi        = phi_neu - phi
      error_height     = height_neu - height
   endwhile

   ; define output vector for one target
   latlonh_pos         = [phi_neu, longitude, height_neu]

endelse
```

End

```
;=============================================================================
;
; Description:
; ------------
; Procedure to calculate the intersection of the antenna pointing vector with
; the Earth ellipsoid.
;
; Inputs:
; -------
; sat_pos_in    - (x,y,z) satellite position vector in inertial system
; sat_vel_in    - (x,y,z) satellite velocity vector in inertial system
; off_nadir     - off-nadir angle [rad]
; squint        - squint angle [rad]
;
; Outputs:
; --------
; target_pos_in - target (x,y,z) position on Earth ellipsoid in inertial system
; range         - range from satellite position to target position
;
; Code maintainer:
; ----------------
; Richard Lord, 25-04-2002
; copyright DLR
;
; References:
; -----------
; X-SAR Geometric Error Budget Analysis, Univ. of Zurich, Nov. 1990, pg. 24
; R.T. Lord and E. Boerner, 'Calculation of Earth Ellipsoid Intersection',
; DLR, July 2002.
;
;=============================================================================


Pro ellipsoidal_intersection, sat_pos_in, sat_vel_in, off_nadir, squint=squint, $
                              target_pos_in=target_pos_in, range=range


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Default values
if (not n_elements(squint)) then squint=0d


;-----Get constants and antenna parameters
nadir_definition = program_execution.nadir_definition
r_equatorial     = constants.semi_major
r_polar          = constants.semi_minor
leftlook         = instrument_input_parameters.leftlook
yaw              = instrument_input_parameters.yaw
roll             = instrument_input_parameters.roll
pitch            = instrument_input_parameters.pitch


;-----Set antenna look direction
if keyword_set(leftlook) then begin
   antenna_direction = -1           ;-1 for left-looking antenna
endif else begin
   antenna_direction = 1            ;+1 for right-looking antenna
endelse
```

```
;-----Error checking
Size_pos = size(sat_pos_in)
Size_vel = size(sat_vel_in)
if (Size_pos[0] ne Size_vel[0]) then begin
   message, 'Error: Position and velocity vectors have different sizes!'
endif


;-----Convert xyz to latlonh
convert_xyz_latlon, sat_pos_in, latlonh=sat_latlonh


;-----Find out if sat_pos and sat_vel are vectors, or arrays of vectors.
if (Size_pos[0] eq 2) then numvectors = Size_pos[1] else numvectors = 1

if (numvectors eq 1) then begin
   sat_pos_in  = reform(sat_pos_in, 1, 3)
   sat_vel_in  = reform(sat_vel_in, 1, 3)
   sat_latlonh = reform(sat_latlonh, 1, 3)
endif

range         = dblarr(numvectors)
target_pos_in = dblarr(numvectors, 3)


n_satellite_system = [cos(off_nadir - (!dpi/2d)) * cos(squint) * antenna_direction, $
                      cos(off_nadir - (!dpi/2d)) * sin(squint), $
                      sin(off_nadir - (!dpi/2d))]

n_trajectory_system = reform(rotz(yaw) ## roty(roll) ## rotx(pitch) ##
n_satellite_system)


for i=0L, (numvectors-1) do begin


   ;-----Find the trajectory system directional unit vectors
   if (nadir_definition eq 'perpendicular') then begin
      zenith = [cos(sat_latlonh[i, 0]) * cos(sat_latlonh[i, 1]), cos(sat_latlonh[i, 0])
* $
                sin(sat_latlonh[i, 1]), sin(sat_latlonh[i, 0])]
   endif else begin
      ;Note: In the following definition of zenith the vector goes through the origin!
      zenith = reform(sat_pos_in[i, *])
   endelse

   xdir = crossp(sat_vel_in[i, *], zenith)
   ydir = crossp(zenith, xdir)
   ez   = zenith / norm(zenith)
   ex   = xdir   / norm(xdir)
   ey   = ydir   / norm(ydir)

   D = [transpose(ex), transpose(ey), transpose(ez)]

   n_inertial_system = reform(D ## n_trajectory_system)


   ;-----A line can be described through a point P=(x1, y1, z1) and a direction
   ;vector r=(a, b, c).  The parametric representation for the line is:
```

```
   ;x = x1 + a*lambda
   ;y = y1 + b*lambda
   ;z = z1 + c*lambda


   ;-----The equation for the earth ellipsoid is:
   ;(x^2 / semimajor^2) + (y^2 / semimajor^2) + (z^2 / semiminor^2) - 1 = 0


   ;-----Now insert the equations for the line into the ellipsoid equation and solve
   ;for lambda.

   a = ((n_inertial_system[0]^2d + n_inertial_system[1]^2d) / (r_equatorial^2d)) + $
       ((n_inertial_system[2]^2d) / r_polar^2d)
   b = (((2d * n_inertial_system[0] * sat_pos_in[i, 0]) + (2d * n_inertial_system[1] * $
       sat_pos_in[i, 1])) / (r_equatorial^2d)) + ((2d * n_inertial_system[2] * $
       sat_pos_in[i, 2]) / (r_polar^2d))
   c = (((sat_pos_in[i, 0]^2d) + (sat_pos_in[i, 1]^2d)) / (r_equatorial^2d)) + $
       ((sat_pos_in[i, 2]^2d) / (r_polar^2d)) - 1d


   sqrt_arg = b^2d - 4*a*c
   if (sqrt_arg ge 0d) then begin
      lambda1 = (-b + sqrt(sqrt_arg)) / (2d * a)
      lambda2 = (-b - sqrt(sqrt_arg)) / (2d * a)

      ;-----We are only interested in the smaller value of lambda, i.e. we don't
      ;want the point where the line exits the ellipsoid.

      lambda = min([lambda1, lambda2])
      range[i] = lambda

      target_pos_in[i, *] = sat_pos_in[i, *] + (n_inertial_system * lambda)

   endif else begin
      print, 'Warning: No real solution for earth trace!'
      target_pos_in[i, *] = [1, 1, 1]  ;To prevent division by 0 when converting to lat/
lon
      range[i]            = -1         ;i.e. an impossible value
   endelse

endfor


if (numvectors eq 1) then begin
   sat_pos_in    = reform(sat_pos_in)
   sat_vel_in    = reform(sat_vel_in)
   target_pos_in = reform(target_pos_in)
   range         = range[0]
endif


End
```

```
;=========================================================
; Description:
; -----------
; Procedure to calculate the fresnel radius.
;
; Inputs:       w               - the wavelength at which the satellite transmit
; -------       n               - the fresnel zone number
;               rec_xyz         - position of the receiver
;               sat_xyz         - satellite position
;               ref_xyz         - reflection position
;
; Output:       f_radius        - fresnel radius [m]
;
; Code maintainer:
; ----------------
; Shikoane Given Phaladi, 18 May 2006
;
;=========================================================



Pro fresnel,w, n, rec_xyz, sat_xyz, ref_xyz, f_radius = temp,starttime=starttime,$
                        timeduration=timeduration, timestep=timestep, timer = timer

; Global variables
@introduce_common_parameters.bat

radeg       = constants.radeg




numpoints  = long(timeduration / timestep) + 1L
tottime    = starttime

temp = dblarr(numpoints)
timer = dblarr(numpoints)

for i=0L, (numpoints-1) do begin

;--------magnitude of the RS and GS vectors

RS = sqrt( (ref_xyz[i,0] - rec_xyz[0])^2d + (ref_xyz[i,1] - rec_xyz[1])^2d )


GS = sqrt( (ref_xyz[i,0] - sat_xyz[i,0])^2d + (ref_xyz[i,1] - sat_xyz[i,1])^2d )
;---------


;---------calculating the Fresnel radius
f_radius = sqrt ((n*w*RS*GS)/(RS + GS))
temp[i,*] = f_radius

timer[i] = tottime
tottime = tottime + timestep



endfor
```

End

```
Pro fresnel_ellipse, final_angle, n, w, sat_pos_ef, rec_pos, ellipse_par = final,
starttime=starttime,$
                        timeduration=timeduration, timestep=timestep


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Find out if xyz_pos is a vector, or an array of vectors.
;size_pos              = size(sat_pos_ef, /N_DIMENSIONS)


;-----Calculate latitude, longitude and height according to WGS84 (iterative
calculation)
;if (size_pos eq 2) then begin
;   numpoints           = n_elements(sat_pos_ef[*, 0])


degrad      = constants.dtor



;-----Now advance until (starttime + timeduration), in timestep intervals
tottime    = starttime
numpoints  = long(timeduration / timestep) + 1L
;ellipse_par            = dblarr(numpoints, 3)
final = dblarr(numpoints)
   for i=0L, (numpoints-1) do begin

   h_rec = rec_pos[2]           ; height of the receiver
   h_gps = sat_pos_ef[i,2]      ; height of the GPS satellite
   r = sqrt ( (rec_pos[0]-sat_pos_ef[i,0])^2d + (rec_pos[1]-sat_pos_ef[i,1])
^2d) ;distance between the rec&sat

   delta_n = n*w/2d
   theta_angle = atan((h_rec - h_gps)/r)*degrad
   sec = 1d/cos(theta_angle)
   term1 = (delta_n/r + sec)^2d
   y1 = (r/2d)*sqrt(((delta_n/r)^2d + 2d*delta_n*sec/r)*(1d - (tan(theta_angle)^2)/
(term1-1d)))
   x1 = y1*sqrt(1d + 1d/(term1 - 1d))

print, y1


endfor


End
```

```
Pro fresnel_ellipse, final_angle, n, w, sat_pos_ef, rec_pos, ellipse_par = final,
starttime=starttime,$
                        timeduration=timeduration, timestep=timestep


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Find out if xyz_pos is a vector, or an array of vectors.
;size_pos              = size(sat_pos_ef, /N_DIMENSIONS)


;-----Calculate latitude, longitude and height according to WGS84 (iterative
calculation)
;if (size_pos eq 2) then begin
;   numpoints           = n_elements(sat_pos_ef[*, 0])


degrad      = constants.dtor



;-----Now advance until (starttime + timeduration), in timestep intervals
tottime    = starttime
numpoints  = long(timeduration / timestep) + 1L
;ellipse_par            = dblarr(numpoints, 3)
final = dblarr(numpoints)
   for i=0L, (numpoints-1) do begin

   h_rec = rec_pos[2]           ; height of the receiver
   h_gps = sat_pos_ef[i,2]      ; height of the GPS satellite
   r = sqrt ( (rec_pos[0]-sat_pos_ef[i,0])^2d + (rec_pos[1]-sat_pos_ef[i,1])
^2d) ;distance between the rec&sat

   delta_n = n*w/2d
   theta_angle = atan((h_rec - h_gps),r)*degrad
   sec = 1d/(cos(theta_angle))
   term1 = (delta_n/r + sec)^2d
   y1 = (r/2d)*sqrt(((delta_n/r)^2d + 2d*delta_n*sec/r)*(1d - ((tan(theta_angle))^2)/
(term1-1d)))
   x1 = y1*sqrt(1d + 1d/(term1 - 1d))
   area = !pi*y1*x1/1000000d

print, y1,x1,area


endfor


End
```

```
;===============================================================================
;                                                                              |
; Description:                                                                 |
; ------------                                                                 |
; Procedure to set global GPS satellite orbit parameters, by reading the gps almanac |
; files from the website "hppt://www.navcen.uscg.gov/ftp/GPS/almanacs/"        |
;                                                                              |
;                                                                              |
; Outputs:                                                                     |
; --------                                                                     |
; orbit parameters [global]: sat_name     []    : satellite name               |
;                            eccentricity []    : orbit eccentricity           |
;                            inclination  [rad] : inclination                  |
;                            semi_major   [m]   : semi major axis of orbit     |
;                            omega        [rad] : angle of perigee             |
;                            asc_node     [rad] : right ascencion of ascending node |
;                            mean_anomaly [rad] : mean anomaly                 |
;                                                                              |
; Code maintainer:                                                             |
; ----------------                                                             |
; Richard Lord, 08-07-2004                                                     |
;                                                                              |
; Updated by:                                                                  |
; -----------                                                                  |
; Shikoane Given Phaladi, 04 May 2006                                          |
;                                                                              |
;===============================================================================


Pro get_gps_orbit_parameters


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Get global parameters
filename    = program_execution.gps_orbit_data
num_gps_ids = n_elements(orbit_parameters.sat_name)


;-----Initialise
sat_name     = strarr(num_gps_ids)
eccentricity = dblarr(num_gps_ids)
inclination  = dblarr(num_gps_ids)
semi_major   = dblarr(num_gps_ids)
omega        = dblarr(num_gps_ids)
asc_node     = dblarr(num_gps_ids)
mean_anomaly = dblarr(num_gps_ids)


;-----Read file
line    = ''
counter = 0L
openr, lun, filename, /get_lun, ERROR=err
if (err eq 0) then begin
   while (not eof(lun)) do begin
      readf, lun, line
      if (strlowcase(strmid(strtrim(line, 2), 0, 3)) eq 'id:') then begin
         sat_name[counter] = strtrim(strmid(line, 3), 2)
         for i=0L, 10 do begin
```

```
            readf, lun, line
            label = strlowcase(strtrim(strmid(line, 0, strpos(line, ':')), 2))
            value = strmid(strmid(line, strpos(line, ':') + 1), 2)
            if (label eq 'eccentricity') then eccentricity[counter] = double(value)
            if (label eq 'orbital inclination(rad)') then inclination[counter] = double
(value)
            if (label eq 'sqrt(a)  (m 1/2)') then semi_major[counter] = double(value) ^
2d
            if (label eq 'right ascen at week(rad)') then asc_node[counter] = double
(value)
            if (label eq 'argument of perigee(rad)') then omega[counter] = double(value)
            if (label eq 'mean anom(rad)') then mean_anomaly[counter] = double(value)
         endfor
         counter = counter + 1L
      endif
   endwhile
   free_lun, lun
endif else begin
   message, 'Error: file ' + filename + ' not found!'
endelse


;----- Update global parameters
orbit_parameters.sat_name     = sat_name
orbit_parameters.eccentricity = eccentricity
orbit_parameters.inclination  = inclination
orbit_parameters.semi_major   = semi_major
orbit_parameters.omega        = omega
orbit_parameters.asc_node     = asc_node
orbit_parameters.mean_anomaly = mean_anomaly


;----- Print information
if (program_execution.verbose_mode eq 1) then begin
   print, strjoin(replicate('-', program_execution.dotted_line))
   print, "Global variables updated by 'set_satellite_orbit_parameters' procedure:"
   print, strjoin(replicate('-', program_execution.dotted_line))
endif


;----- Adjust semi-major axis to get an exact 1 day repeat cycle
if (program_execution.adjust_semimajor eq 1) then begin
   for i=0L, (num_gps_ids - 1L) do begin
      if (orbit_parameters.sat_name[i] ne '') then get_repeat_cycle_semimajor, i
   endfor
endif



End
```

```
;=====================================================================
;
; Description:
; ------------
; Procedure to modify the semi-major axis corresponding to the repeat
; cycle specified by the number of days.
;
; Code maintainer:
; ----------------
; Richard Lord, 09-07-2004
;
;
; Reviewed by:
; ------------
; Shikoane G. Phaladi, April-2006
;
;
; Note: One repeat cycle is *less* than 24 hours, and is roughly 26h56min
;
;=====================================================================


Pro get_repeat_cycle_semimajor, sat_num


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Get constants and parameters
radeg          = constants.radeg
gm_earth       = constants.gm_earth
omega_earth    = constants.omega_earth
omega          = orbit_parameters.omega[sat_num]
semi_major     = orbit_parameters.semi_major[sat_num]
old_semi_major = orbit_parameters.semi_major[sat_num]
days           = program_execution.repeat_cycle_days


;-----User-defined parameters
max_delta_lon = 0.1d
max_delta_lat = 1d-12


;-----Some calculations
single_day    = (2d * !dpi) / omega_earth
timeduration  = 0d
timestep      = 1d


;-----Change omega temporarily to zero if it is non-zero, because
;     this procedure is designed for omega=zero
orbit_parameters.omega[sat_num] = 0d


;-----Get satellite statevectors in inertial system at time=0
starttime = 0d
satellite_inertial_vectors, sat_pos_in, sat_vel_in, sat_num, starttime=starttime, $
                            timeduration=timeduration, timestep=timestep
transform_inertial_to_fixed, sat_pos_in, pos_ef=sat_pos_ef_start, starttime=starttime, $
                             timeduration=timeduration, timestep=timestep
```

```
convert_xyz_latlon, sat_pos_ef_start, latlonh=sat_latlonh_start


;-----Get satellite statevectors in inertial system at time=repeat-cycle time
starttime = single_day * days
satellite_inertial_vectors, sat_pos_in, sat_vel_in, sat_num, starttime=starttime, $
                            timeduration=timeduration, timestep=timestep
transform_inertial_to_fixed, sat_pos_in, pos_ef=sat_pos_ef_end, starttime=starttime, $
                             timeduration=timeduration, timestep=timestep
convert_xyz_latlon, sat_pos_ef_end, latlonh=sat_latlonh_end


;-----Increase semi-major until longitude corresponds to start value
delta_lon        = sat_latlonh_end[1] - sat_latlonh_start[1]
delta_lon_orig   = delta_lon
delta_semimajor1 = 0d
while (abs(delta_lon) gt max_delta_lon) do begin
   orbit_parameters.semi_major[sat_num] = semi_major + delta_semimajor1
   satellite_inertial_vectors, sat_pos_in, sat_vel_in, sat_num, starttime=starttime, $
                               timeduration=timeduration, timestep=timestep
   transform_inertial_to_fixed, sat_pos_in, pos_ef=sat_pos_ef_end, starttime=starttime,
$
                                timeduration=timeduration, timestep=timestep
   convert_xyz_latlon, sat_pos_ef_end, latlonh=sat_latlonh_end
   delta_semimajor1 = delta_semimajor1 + 10d
   delta_lon        = sat_latlonh_end[1] - sat_latlonh_start[1]
endwhile


;-----Decrease semi-major until longitude corresponds to start value
delta_lon        = delta_lon_orig
delta_semimajor2 = 0d
while (abs(delta_lon) gt max_delta_lon) do begin
   orbit_parameters.semi_major[sat_num] = semi_major + delta_semimajor2
   satellite_inertial_vectors, sat_pos_in, sat_vel_in, sat_num, starttime=starttime, $
                               timeduration=timeduration, timestep=timestep
   transform_inertial_to_fixed, sat_pos_in, pos_ef=sat_pos_ef_end, starttime=starttime,
$
                                timeduration=timeduration, timestep=timestep
   convert_xyz_latlon, sat_pos_ef_end, latlonh=sat_latlonh_end
   delta_semimajor2 = delta_semimajor2 - 10d
   delta_lon        = sat_latlonh_end[1] - sat_latlonh_start[1]
endwhile


;-----Now choose semi-major which is closest to original semi-major
if (delta_semimajor1 lt abs(delta_semimajor2)) then begin
   orbit_parameters.semi_major[sat_num] = semi_major + delta_semimajor1
endif else begin
   orbit_parameters.semi_major[sat_num] = semi_major + delta_semimajor2
endelse


;-----Now fine-tune result
satellite_inertial_vectors, sat_pos_in, sat_vel_in, sat_num, starttime=starttime, $
                            timeduration=timeduration, timestep=timestep
transform_inertial_to_fixed, sat_pos_in, pos_ef=sat_pos_ef_end, starttime=starttime, $
                             timeduration=timeduration, timestep=timestep
convert_xyz_latlon, sat_pos_ef_end, latlonh=sat_latlonh_end

delta_lat1 = sat_latlonh_end[0] - sat_latlonh_start[0]
```

```
delta_semimajor = 1d

while (abs(delta_lat1) gt max_delta_lat) do begin
   orbit_parameters.semi_major[sat_num] = orbit_parameters.semi_major[sat_num] +
delta_semimajor
   satellite_inertial_vectors, sat_pos_in, sat_vel_in, sat_num, starttime=starttime, $
                               timeduration=timeduration, timestep=timestep
   transform_inertial_to_fixed, sat_pos_in, pos_ef=sat_pos_ef_end, starttime=starttime,
$
                                timeduration=timeduration, timestep=timestep
   convert_xyz_latlon, sat_pos_ef_end, latlonh=sat_latlonh_end

   delta_lat2      = sat_latlonh_end[0] - sat_latlonh_start[0]
   delta_semimajor = (delta_semimajor / (delta_lat1 - delta_lat2)) * delta_lat2
   delta_lat1      = delta_lat2
endwhile


;-----Change omega back to original value
orbit_parameters.omega[sat_num] = omega


;-----Calculate number of orbits during repeat cycle
period        = 2d * !dpi * sqrt((orbit_parameters.semi_major[sat_num]^3d) / gm_earth)
numorbits     = (single_day * days) / period


;----- Print information
if (program_execution.verbose_mode eq 1) then begin
   print, strjoin(replicate('-', program_execution.dotted_line))
   print, "Global variables updated by 'get_repeat_cycle_semimajor' procedure:"
   print, ' '
   print, 'GPS satellite ID                                    = ' + $
          strtrim(orbit_parameters.sat_name[sat_num], 2)
   print, 'previous value for orbit_parameters.semi_major [km] = ' + $
          strtrim(old_semi_major / 1000d, 2)
   print, 'new value for orbit_parameters.semi_major [km]      = ' + $
          strtrim(orbit_parameters.semi_major[sat_num] / 1000d, 2)
   print, 'Repeat cycle [days]                                 = ' + strtrim(days, 2)
   print, 'Number of orbits during repeat cycle                = ' + strtrim(numorbits,
2)
   print, strjoin(replicate('-', program_execution.dotted_line))
endif


End
```

```
******** Week  349 almanac for PRN-01 ********
ID:                         01
Health:                     000
Eccentricity:               0.6185531616E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9878201110
Rate of Right Ascen(r/s):  -0.7726036106E-008
SQRT(A)  (m 1/2):           5153.645996
Right Ascen at Week(rad):   0.2938452926E+001
Argument of Perigee(rad):  -1.721084874
Mean Anom(rad):             0.1485957003E+001
Af0(s):                     0.5149841309E-004
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-02 ********
ID:                         02
Health:                     000
Eccentricity:               0.9124279022E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9506390532
Rate of Right Ascen(r/s):  -0.8046049436E-008
SQRT(A)  (m 1/2):           5153.620605
Right Ascen at Week(rad):   0.8026262357E+000
Argument of Perigee(rad):   2.045684721
Mean Anom(rad):             0.2886735378E+001
Af0(s):                    -0.4768371582E-005
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-03 ********
ID:                         03
Health:                     000
Eccentricity:               0.8007049561E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9264968322
Rate of Right Ascen(r/s):  -0.8148910863E-008
SQRT(A)  (m 1/2):           5153.622070
Right Ascen at Week(rad):  -0.3330457297E+000
Argument of Perigee(rad):   0.660341278
Mean Anom(rad):             0.2079763737E+001
Af0(s):                     0.9632110596E-004
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-04 ********
ID:                         04
Health:                     000
Eccentricity:               0.7346630096E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9501716685
Rate of Right Ascen(r/s):  -0.8068907531E-008
SQRT(A)  (m 1/2):           5153.524414
Right Ascen at Week(rad):   0.8223249309E+000
Argument of Perigee(rad):   0.153832881
Mean Anom(rad):            -0.1022862209E+001
Af0(s):                     0.2231597900E-003
Af1(s/s):                   0.1091393642E-010
week:                        349

******** Week  349 almanac for PRN-05 ********
```

```
ID:                         05
Health:                     000
Eccentricity:               0.6904602051E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9375163270
Rate of Right Ascen(r/s):  -0.8034620388E-008
SQRT(A)  (m 1/2):           5153.549805
Right Ascen at Week(rad):  -0.1371249245E+001
Argument of Perigee(rad):   1.039119934
Mean Anom(rad):            -0.1275164224E+001
Af0(s):                    -0.4959106445E-004
Af1(s/s):                   0.5820766091E-010
week:                        349

******** Week  349 almanac for PRN-06 ********
ID:                         06
Health:                     000
Eccentricity:               0.6074428558E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9340528860
Rate of Right Ascen(r/s):  -0.8057478483E-008
SQRT(A)  (m 1/2):           5153.728516
Right Ascen at Week(rad):  -0.2746675741E+000
Argument of Perigee(rad):  -1.834447028
Mean Anom(rad):            -0.1169585449E-001
Af0(s):                     0.5102157593E-003
Af1(s/s):                   0.1818989404E-010
week:                        349

******** Week  349 almanac for PRN-07 ********
ID:                         07
Health:                     000
Eccentricity:               0.1061677933E-001
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9355748825
Rate of Right Ascen(r/s):  -0.8023191341E-008
SQRT(A)  (m 1/2):           5153.783691
Right Ascen at Week(rad):  -0.3007774551E+000
Argument of Perigee(rad):  -1.839791617
Mean Anom(rad):            -0.9269273654E+000
Af0(s):                     0.5350112915E-003
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-08 ********
ID:                         08
Health:                     000
Eccentricity:               0.9479522705E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9727559403
Rate of Right Ascen(r/s):  -0.7897471819E-008
SQRT(A)  (m 1/2):           5153.585449
Right Ascen at Week(rad):  -0.2295981625E+001
Argument of Perigee(rad):   2.652193610
Mean Anom(rad):             0.9108759942E-002
Af0(s):                    -0.6484985352E-004
Af1(s/s):                   0.0000000000E+000
week:                        349

******** Week  349 almanac for PRN-09 ********
ID:                         09
```

```
Health:                      000
Eccentricity:                0.1782655716E-001
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):    0.9603582596
Rate of Right Ascen(r/s):  -0.8000333246E-008
SQRT(A)  (m 1/2):            5153.581543
Right Ascen at Week(rad):  -0.2375087995E+001
Argument of Perigee(rad):    1.230949549
Mean Anom(rad):            -0.3047588393E+000
Af0(s):                      0.1907348633E-004
Af1(s/s):                    0.0000000000E+000
week:                         349

******** Week  349 almanac for PRN-10 ********
ID:                          10
Health:                      000
Eccentricity:                0.7140159607E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):    0.9739004338
Rate of Right Ascen(r/s):  -0.7737465154E-008
SQRT(A)  (m 1/2):            5153.562012
Right Ascen at Week(rad):    0.1873777007E+001
Argument of Perigee(rad):    0.433970505
Mean Anom(rad):            -0.3046721035E+001
Af0(s):                      0.8296966553E-004
Af1(s/s):                    0.0000000000E+000
week:                         349

******** Week  349 almanac for PRN-11 ********
ID:                          11
Health:                      000
Eccentricity:                0.5408287048E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):    0.8977886214
Rate of Right Ascen(r/s):  -0.8583214668E-008
SQRT(A)  (m 1/2):            5153.624023
Right Ascen at Week(rad):    0.6592061477E+000
Argument of Perigee(rad):    0.375468762
Mean Anom(rad):              0.5243656412E+000
Af0(s):                      0.3252029419E-003
Af1(s/s):                    0.3637978807E-011
week:                         349

******** Week  349 almanac for PRN-13 ********
ID:                          13
Health:                      000
Eccentricity:                0.2954959869E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):    0.9920026055
Rate of Right Ascen(r/s):  -0.7668890869E-008
SQRT(A)  (m 1/2):            5153.508789
Right Ascen at Week(rad):    0.2923989839E+001
Argument of Perigee(rad):    1.138345197
Mean Anom(rad):            -0.3044524177E+001
Af0(s):                      0.6008148193E-004
Af1(s/s):                    0.3637978807E-011
week:                         349

******** Week  349 almanac for PRN-14 ********
ID:                          14
Health:                      000
```

```
Eccentricity:               0.2295970917E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9861662880
Rate of Right Ascen(r/s):  -0.7726036106E-008
SQRT(A)  (m 1/2):           5153.600586
Right Ascen at Week(rad):   0.2910962237E+001
Argument of Perigee(rad):  -2.023092959
Mean Anom(rad):             0.2446149580E+001
Af0(s):                    -0.1430511475E-004
Af1(s/s):                   0.0000000000E+000
week:                        349

******** Week  349 almanac for PRN-15 ********
ID:                         15
Health:                     000
Eccentricity:               0.9704113007E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9571884321
Rate of Right Ascen(r/s):  -0.7943188009E-008
SQRT(A)  (m 1/2):           5153.641113
Right Ascen at Week(rad):   0.8795604655E+000
Argument of Perigee(rad):   2.602870660
Mean Anom(rad):             0.5499148851E-001
Af0(s):                     0.6074905396E-003
Af1(s/s):                   0.7275957614E-011
week:                        349

******** Week  349 almanac for PRN-16 ********
ID:                         16
Health:                     000
Eccentricity:               0.3489494324E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9618622799
Rate of Right Ascen(r/s):  -0.7748894201E-008
SQRT(A)  (m 1/2):           5153.658203
Right Ascen at Week(rad):  -0.1280666106E+001
Argument of Perigee(rad):  -0.899396605
Mean Anom(rad):            -0.1439752198E+001
Af0(s):                     0.4100799561E-004
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-17 ********
ID:                         17
Health:                     000
Eccentricity:               0.1974582672E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9605020703
Rate of Right Ascen(r/s):  -0.7794610391E-008
SQRT(A)  (m 1/2):           5153.627930
Right Ascen at Week(rad):  -0.2418367900E+000
Argument of Perigee(rad):   2.970415228
Mean Anom(rad):            -0.2488477114E+001
Af0(s):                     0.8964538574E-004
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-18 ********
ID:                         18
Health:                     000
Eccentricity:               0.6961822510E-002
```

```
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9579434383
Rate of Right Ascen(r/s):  -0.7908900866E-008
SQRT(A)  (m 1/2):           5153.632812
Right Ascen at Week(rad):   0.1895877041E+001
Argument of Perigee(rad):  -2.696505656
Mean Anom(rad):            -0.1497229664E+001
Af0(s):                    -0.2336502075E-003
Af1(s/s):                   0.0000000000E+000
week:                        349

******** Week  349 almanac for PRN-19 ********
ID:                         19
Health:                     000
Eccentricity:               0.3419876099E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9585126890
Rate of Right Ascen(r/s):  -0.7840326581E-008
SQRT(A)  (m 1/2):           5153.544434
Right Ascen at Week(rad):  -0.1823332408E+000
Argument of Perigee(rad):  -1.301625018
Mean Anom(rad):            -0.2837598558E+001
Af0(s):                    -0.1239776611E-004
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-20 ********
ID:                         20
Health:                     000
Eccentricity:               0.2979278564E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9575060141
Rate of Right Ascen(r/s):  -0.7897471819E-008
SQRT(A)  (m 1/2):           5153.722168
Right Ascen at Week(rad):   0.1842970059E+001
Argument of Perigee(rad):   1.342084510
Mean Anom(rad):            -0.1689822900E+001
Af0(s):                    -0.2765655518E-004
Af1(s/s):                   0.0000000000E+000
week:                        349

******** Week  349 almanac for PRN-21 ********
ID:                         21
Health:                     000
Eccentricity:               0.1110792160E-001
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9437361397
Rate of Right Ascen(r/s):  -0.8080336578E-008
SQRT(A)  (m 1/2):           5153.660156
Right Ascen at Week(rad):   0.8426868780E+000
Argument of Perigee(rad):  -2.993878093
Mean Anom(rad):            -0.2116294276E+000
Af0(s):                     0.2670288086E-004
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-22 ********
ID:                         22
Health:                     000
Eccentricity:               0.4662036896E-002
Time of Applicability(s):  503808.0000
```

```
Orbital Inclination(rad):   0.9554746880
Rate of Right Ascen(r/s):  -0.7943188009E-008
SQRT(A)  (m 1/2):           5153.633789
Right Ascen at Week(rad):   0.1904049533E+001
Argument of Perigee(rad):  -1.602441796
Mean Anom(rad):             0.3085494871E+001
Af0(s):                     0.9250640869E-004
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-23 ********
ID:                         23
Health:                     000
Eccentricity:               0.4391670227E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9683936824
Rate of Right Ascen(r/s):  -0.7908900866E-008
SQRT(A)  (m 1/2):           5153.522461
Right Ascen at Week(rad):   0.2897796817E+001
Argument of Perigee(rad):   2.423567930
Mean Anom(rad):             0.2496570211E+001
Af0(s):                     0.1420974731E-003
Af1(s/s):                   0.0000000000E+000
week:                        349

******** Week  349 almanac for PRN-24 ********
ID:                         24
Health:                     000
Eccentricity:               0.8998394012E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9603942123
Rate of Right Ascen(r/s):  -0.7943188009E-008
SQRT(A)  (m 1/2):           5153.797852
Right Ascen at Week(rad):   0.8558781391E+000
Argument of Perigee(rad):  -0.930583303
Mean Anom(rad):             0.5902320648E+000
Af0(s):                     0.3814697266E-005
Af1(s/s):                   0.3637978807E-011
week:                        349

******** Week  349 almanac for PRN-25 ********
ID:                         25
Health:                     000
Eccentricity:               0.1262235641E-001
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9537309833
Rate of Right Ascen(r/s):  -0.8103194673E-008
SQRT(A)  (m 1/2):           5153.695312
Right Ascen at Week(rad):  -0.2429478400E+001
Argument of Perigee(rad):  -1.352625759
Mean Anom(rad):             0.3259035062E+000
Af0(s):                     0.9536743164E-005
Af1(s/s):                  -0.7275957614E-011
week:                        349

******** Week  349 almanac for PRN-26 ********
ID:                         26
Health:                     000
Eccentricity:               0.1735305786E-001
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9901570349
```

```
Rate of Right Ascen(r/s):  -0.7691748964E-008
SQRT(A)  (m 1/2):           5153.635254
Right Ascen at Week(rad):   0.2924841468E+001
Argument of Perigee(rad):   0.775613044
Mean Anom(rad):             0.1528048222E+001
Af0(s):                    -0.1068115234E-003
Af1(s/s):                  -0.1091393642E-010
week:                       349

******** Week  349 almanac for PRN-27 ********
ID:                         27
Health:                     000
Eccentricity:               0.1978826523E-001
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9582490361
Rate of Right Ascen(r/s):  -0.8034620388E-008
SQRT(A)  (m 1/2):           5153.598145
Right Ascen at Week(rad):  -0.2397764770E+001
Argument of Perigee(rad):  -1.926995954
Mean Anom(rad):            -0.1211073712E+001
Af0(s):                     0.4768371582E-004
Af1(s/s):                   0.0000000000E+000
week:                       349

******** Week  349 almanac for PRN-28 ********
ID:                         28
Health:                     000
Eccentricity:               0.1139259338E-001
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9599268275
Rate of Right Ascen(r/s):  -0.7760323249E-008
SQRT(A)  (m 1/2):           5153.617676
Right Ascen at Week(rad):  -0.1271032288E+001
Argument of Perigee(rad):  -2.252512594
Mean Anom(rad):            -0.2373315079E+001
Af0(s):                     0.2765655518E-004
Af1(s/s):                   0.0000000000E+000
week:                       349

******** Week  349 almanac for PRN-29 ********
ID:                         29
Health:                     000
Eccentricity:               0.9394645691E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9868254203
Rate of Right Ascen(r/s):  -0.7726036106E-008
SQRT(A)  (m 1/2):           5153.668457
Right Ascen at Week(rad):   0.2890304804E+001
Argument of Perigee(rad):  -0.910185403
Mean Anom(rad):            -0.2893276517E+001
Af0(s):                     0.5989074707E-003
Af1(s/s):                   0.7275957614E-011
week:                       349

******** Week  349 almanac for PRN-30 ********
ID:                         30
Health:                     000
Eccentricity:               0.8832454681E-002
Time of Applicability(s):  503808.0000
Orbital Inclination(rad):   0.9441256270
Rate of Right Ascen(r/s):  -0.7966046104E-008
```

```
SQRT(A)  (m 1/2):           5153.702148
Right Ascen at Week(rad):  -0.1324868422E+001
Argument of Perigee(rad):   1.290583427
Mean Anom(rad):            -0.2036445258E+001
Af0(s):                     0.2193450928E-003
Af1(s/s):                   0.3274180926E-010
week:                        349
```

```
COMMON gps, constants, instrument_input_parameters, instrument_calculated_parameters,
orbit_parameters, scene_parameters , scene_calculated_parameters, target_positions,
program_execution
```

```
;=====================================================================
;
; Description:
; ------------
; Procedure to plot longitude and latitude coordinates on a world map.
;
; Inputs:
; -------
; lon_arr        - Array of longitude values [rad]
; lat_arr        - Array of latitude values [rad]
;
; Keywords:
; ---------
; ortho          - Orthogonal projection, as opposed to default mercator
; cylindrical    - Cylindrical projection
; keepoldplot    - Plot on existing window
; earthtrace     - If set, plot is drawn in a different colour
; zoom           - Zoom in by setting limits on plot window
; hires          - Draw high-resolution map
; rivers         - Draw rivers
; grid           - Draw gridlines
; margin_north   - A factor by which the northern plot limit is increased
; margin_south   - A factor by which the southern plot limit is increased
; margin_east    - A factor by which the eastern plot limit is increased
; margin_west    - A factor by which the western plot limit is increased
; psym           - Symbol used for plotting target or satellite 'head'
;                  psym=8 uses a user-defined symbol, which is a circle
;
; Output:
; -------
; Plot on a world map.
;
; Code maintainer:
; ----------------
; Richard Lord, 08-07-2004
;
; Updated by:
; -----------
; Shikoane Given Phaladi, May 2006
;=====================================================================


Pro plot_map, lon_arr=lon_arr, $
              lat_arr=lat_arr, $
              cylindrical=cylindrical, $
              keepoldplot=keepoldplot, $
              clean=clean, $
              earthtrace=earthtrace, $
              hires=hires, $
              rivers=rivers, $
              grid = grid, $
              margin_north=margin_north, $
              margin_south=margin_south, $
              margin_east=margin_east, $
              margin_west=margin_west, $
              color = color,$
              psym = psym


radeg = 180d / !dpi
```

```
if (not keyword_set(hires)) then hires=0
if (not keyword_set(rivers)) then rivers=0
if (not keyword_set(grid)) then grid=0
if (not n_elements(margin_north)) then margin_north=0.8
if (not n_elements(margin_south)) then margin_south=0.8
if (not n_elements(margin_east)) then margin_east=0.8
if (not n_elements(margin_west)) then margin_west=0.8
if (not n_elements(psym)) then psym=8
if (not n_elements(color)) then COLOR='00FFFF'x

device, decomposed=0

ocean_col     = 0
grid_col      = 1
continent_col = 2
coast_col     = 3
countries_col = 4
rivers_col    = 5
orbit_col     = 6
trace_col     = 7
frame_col     = 255
ref_col = 8
rec_col = 9

tvlct, 177, 230, 241, ocean_col
tvlct,   0,   0,   0, grid_col
tvlct, 250, 235, 215, continent_col
tvlct, 255,   0,   0, coast_col
tvlct, 255,   0,   0, countries_col
tvlct,   0,   0, 255, rivers_col
tvlct, 255,   0, 255, orbit_col
tvlct,  46, 139,  87, trace_col
tvlct,   0,   0,   0, frame_col
tvlct,   0, 255,   0, ref_col
tvlct,   0,   0, 255, rec_col


lat_center = 0.0
lon_center = 0.0
Limit = [-20.0, 5.0, -50.0, 45.0]
;Limit = [-90.0, -180.0, 90.0, 180.0]

if (not keyword_set(keepoldplot)) then begin
   window, /free, xsize=740, ysize=800
   if (keyword_set(cylindrical)) then begin
      map_set, lat_center, lon_center, Limit=Limit, /cylindrical, /isotropic
   endif else begin
      map_set, lat_center, lon_center, Limit=Limit, /mercator, /isotropic
   endelse
   map_continents, hires=hires, /fill_continents, color=continent_col
   map_continents, hires=hires, /coasts, color=coast_col
   map_continents, hires=hires, /countries, color=countries_col
   if (rivers) then begin
      map_continents, hires=hires, /rivers, color=rivers_col
   endif
   if (grid) then begin
      map_grid, color=grid_col, /label
   endif
endif

if (keyword_set(clean)) then begin
```

```
   if (keyword_set(cylindrical)) then begin
      map_set, lat_center, lon_center, Limit=Limit, /cylindrical, /isotropic
   endif else begin
      map_set, lat_center, lon_center, Limit=Limit, /mercator, /isotropic
   endelse
   map_continents, hires=hires, /fill_continents, color=continent_col
   map_continents, hires=hires, /coasts, color=coast_col
   map_continents, hires=hires, /countries, color=countries_col
   if (rivers) then begin
      map_continents, hires=hires, /rivers, color=rivers_col
   endif
   if (grid) then begin
      map_grid, color=grid_col, /label
   endif
endif

;s = 1
;thick = randomu(s)
thick = 2
;fill  = 1
qN    = 49     ;number of points
qr    = 2.0    ;radius
qa    = (findgen(qN) / (qN - 1.0)) * 2d * !dpi
qx    = qr * sin(qa)
qy    = qr * cos(qa)
usersym, qx, qy, /fill ;thick=thick,




if (n_elements(lon_arr)) then begin
   numpoints = n_elements(lon_arr)
   if (keyword_set(earthtrace)) then begin
      oplot, lon_arr * radeg, lat_arr * radeg, color=trace_col, thick=1
   endif else begin
      oplot, lon_arr * radeg, lat_arr * radeg, color=orbit_col, thick=2

      oplot, lon_arr[numpoints-1:numpoints-1] * radeg, $
             lat_arr[numpoints-1:numpoints-1] * radeg, color=orbit_col, thick=2,psym =
8

      ;oplot, lon_arr[numpoints-1:numpoints-1] * radeg, $
      ;      lat_arr[numpoints-1:numpoints-1] * radeg, color=ref_col, thick=1, psym = 1



      ;oplot, lon_arr[numpoints-1:numpoints-1] * radeg, $
        ;     lat_arr[numpoints-1:numpoints-1] * radeg, color=rec_col, thick=1, psym = 3

   endelse
endif


End
```

```
;=====================================================================
;
; Description:
; ------------
; Procedure to calculate the signal reflection position.
;
; Inputs:
; -------
; starttime     [s]   : PRF-synchronized starting time of orbit observation
; timeduration  [s]   : duration of orbit observation
; timestep      [s]   : time between statevector outputs
; sat_pos_ef    [xyz] : position vector in earth-fixed system
; rec_pos       [xyz] : position vector of the receiver
;
; Output:
; ------
; ref_xyz_pos   [xyz] : reflection position
;
; Code Maintainer:
; ----------------
; Shikoane Given Phaladi, 08-May-2006
;
;=====================================================================


Pro reflection_pos, sat_pos_ef, rec_pos, ref_xyz_pos = ref_xyz_pos, starttime=starttime,
$
                        timeduration=timeduration, timestep=timestep


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Find out if xyz_pos is a vector, or an array of vectors.
;size_pos              = size(sat_pos_ef, /N_DIMENSIONS)


;-----Calculate latitude, longitude and height according to WGS84 (iterative
calculation)
;if (size_pos eq 2) then begin
;   numpoints          = n_elements(sat_pos_ef[*, 0])




;-----Now advance until (starttime + timeduration), in timestep intervals

numpoints  = long(timeduration / timestep) + 1L
;ref_xyz_ini         = dblarr(numpoints, 3)
ref_xyz_pos         = dblarr(numpoints, 3)
tottime    = starttime
   for i=0L, (numpoints-1) do begin

   h_rec = rec_pos[2]           ; height of the receiver
   h_gps = sat_pos_ef[i,2]      ; height of the GPS satellite

   diff = [(sat_pos_ef[i,0] - rec_pos[0]),(sat_pos_ef[i,1] - rec_pos[1]),$
   (sat_pos_ef[i,2] - rec_pos[2])]
```

```
   ref_xyz_pos[i,*] = rec_pos + (h_rec/(h_rec+h_gps))*diff



   tottime = tottime + timestep

   endfor

;if (numpoints eq 1) then begin
;   ref_xyz_pos = reform(ref_xyz_pos)
;endif



End
```

```
;=============================================================
;
; Description:
; ------------
; Function which returns the rotation matrix around the x-axis.
;
; Code maintainer:
; ----------------
; Richard Lord, 17-04-2002
; copyright DLR
;
; Reviewed by:
; ------------
; Shikoane G. Phaladi, April-2006
;
;=============================================================


Function rotx, angle

return, [[1d,         0d,          0d], $
         [0d, cos(angle), -sin(angle)], $
         [0d, sin(angle),  cos(angle)]]

End
```

```
;=====================================================================
;
; Description:
; ------------
; Function which returns the rotation matrix around the y-axis.
;
; Code maintainer:
; ----------------
; Richard Lord, 17-04-2002
; copyright DLR
;
; Reviewed by:
; ------------
; Shikoane G. Phaladi, April-2006
;
;=====================================================================


Function roty, angle

return, [[ cos(angle), 0d, sin(angle)], $
         [          0d, 1d,          0d], $
         [-sin(angle), 0d, cos(angle)]]

End
```

```
;=====================================================================
;
; Description:
; ------------
; Function which returns the rotation matrix around the z-axis.
;
; Code maintainer:
; ----------------
; Richard Lord, 17-04-2002
; copyright DLR
;
; Reviewed by:
; ------------
; Shikoane G. Phaladi, April-2006
;
;=====================================================================


Function rotz, angle

return, [[cos(angle), -sin(angle), 0d], $
         [sin(angle),  cos(angle), 0d], $
         [        0d,          0d, 1d]]

End
```

```
;=====================================================================
;
; Description:
; ------------
; Procedure to calculate satellite position and velocity vectors in the
; inertial system.
;
; Inputs:
; -------
; starttime     [s]   : PRF-synchronized starting time of orbit observation
; timeduration  [s]   : duration of orbit observation
; timestep      [s]   : time between statevector outputs
; printon       []    : if set, interesting orbit parameters are printed
; [global]: gm_earth            : gravitation constant * mass of earth
;           r_equatorial  [m]   : equatorial radius of earth
;           eccentricity  []    : eccentricity
;           semi_major    [m]   : semi major axis of orbit
;           inclination   [rad] : inclination of orbit
;           asc_node      [rad] : right ascencion of ascending node
;           omega         [rad] : angle of perigee
;           mean_anomaly  [rad] : mean anomaly
;
; Outputs:
; --------
; sat_pos_in    [m]   : position vectors of partial orbit in earth-fixed system
; sat_vel_in    [m/s] : velocity vectors of partial orbit in earth-fixed system
;
; Note:
; -----
; Time zero (t0) is defined when the satellite passes through the perigee.
;
; Code maintainer:
; ----------------
; Richard Lord, 24-04-2002
; copyright DLR
;
; Updated and modified by:
; ------------------------
; Shikoane Given Phaladi, 04 May 2006
;
; References:
; -----------
; Matlab orbits.m procedure
; Seeber, G. "Satellite Geodesy", de Gruyter, 1993.
; Montenbruck, O. and Gill, E. "Satellite Orbits: Models, Methods, and
;                               applications", Springer, 2000.
; Lord, R.T. "Keplerian Orbit Simulation in an Earth-fixed Terrestrial
; Reference System", DLR, July 2002.
;
;=====================================================================


Pro satellite_inertial_vectors, sat_pos_in, sat_vel_in, sat_num, starttime=starttime, $
                      timeduration=timeduration, timestep=timestep, printon=printon


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Setting the accuracy to use in Newton's Method
```

```
newtons_method_accuracy = 1d-12



;-----Get Earth parameters based on Earth-Model WGS84 and JGM3.
gm_earth     = constants.gm_earth       ; gravitation constant * mass of earth
r_equatorial = constants.semi_major     ; earth radius at equator [m]
radeg = constants.radeg                 ; exact convention from radians to degrees



;-----Get orbit parameters
eccentricity = orbit_parameters.eccentricity[sat_num]
semi_major   = orbit_parameters.semi_major[sat_num]
inclination  = orbit_parameters.inclination[sat_num]
asc_node     = orbit_parameters.asc_node[sat_num]
omega        = orbit_parameters.omega[sat_num]
;mean_anomaly = orbit_parameters.omega[sat_num]


;-----Default values for keywords
if (not n_elements(starttime)) then starttime=0d
if (not n_elements(timeduration)) then timeduration=6000d
if (not n_elements(timestep)) then timestep=10d


;-----Error checking
if (timestep le 0d) then begin
   timestep = abs(timestep) > 1.0
   print, 'Warning: Inappropriate value for timestep.'
   print, '         Setting timestep to ', strtrim(timestep, 2)
endif
if (timeduration lt 0d) then begin
   timeduration = abs(timeduration) > 0d
   print, 'Warning: Setting timeduration to ', strtrim(timeduration)
endif


;-----Orbit parameter calculations

;__________The condition for getting the perigee
if (eccentricity ge 1d) then begin
   print, 'Warning: The perigee value is infinity'
endif

if ((eccentricity ge 0d) and (eccentricity lt 1d)) then begin
   perigee      = semi_major * (1d - eccentricity)
endif
;_________end of the condition

apogee       = semi_major * (1d + eccentricity)
satmaxheight = apogee - r_equatorial
satminheight = perigee - r_equatorial
semiminor    = semi_major * sqrt(1d - (eccentricity^2d))
totarea      = !dpi * semi_major * semiminor
vel_circ     = sqrt(gm_earth / semi_major)
period       = 2d * !dpi * semi_major / vel_circ
mean_motion  = 2d * !dpi / period
sqrt_factor1 = sqrt(1d - (eccentricity^2d))
```

```
;----Print interesting parameters
if keyword_set(printon) then begin
   print, ''
   print, 'Sat min height = ', strtrim(satminheight/1000d, 2), ' [km]'
   print, 'Sat max height = ', strtrim(satmaxheight/1000d, 2), ' [km]'
   print, 'Perigee        = ', strtrim(perigee/1000d, 2), ' [km]'
   print, 'Apogee         = ', strtrim(apogee/1000d, 2), ' [km]'
   print, 'Eccentricity   = ', strtrim(eccentricity, 2)
   print, 'Semimajor Axis = ', strtrim(semi_major/1000d, 2), ' [km]'
   print, 'Semiminor Axis = ', strtrim(semiminor/1000d, 2), ' [km]'
   print, 'Period         = ', strtrim(period, 2), ' [s]'
   print, 'Mean motion    = ', strtrim(mean_motion, 2), ' [rad/s]'
   print, 'Mean Anomaly   = ', strtrim(mean_anomaly,2), ' [rad]'
   print, ''
endif


;-----Now advance until (starttime + timeduration), in timestep intervals
tottime    = starttime
numpoints  = long(timeduration / timestep) + 1L
sat_pos_in = dblarr(numpoints, 3)
sat_vel_in = dblarr(numpoints, 3)

for i=0L, (numpoints-1) do begin
   mean_anomaly = (mean_motion * tottime) mod (2d * !dpi)              ; I'm not sure
if I do agree with u with this
   if (eccentricity ge 0.8) then Ep = !dpi else Ep = mean_anomaly
   deltaE = 1d
   while (deltaE gt newtons_method_accuracy) do begin
      E = Ep - ((Ep - (eccentricity * sin(Ep)) - mean_anomaly) / (1d - (eccentricity *
cos(Ep))))
      deltaE = abs(E - Ep)
      Ep = E
   endwhile

   radius = semi_major * (1d - (eccentricity * cos(E)))
   factor2 = mean_motion * (semi_major^2) / radius
   Rotation = Rotz(asc_node) ## Rotx(inclination) ## Rotz(omega)

   sat_pos_in[i, *] = Rotation ## (semi_major * [cos(E) - eccentricity, sqrt_factor1 *
sin(E) , 0d])
   sat_vel_in[i, *] = Rotation ## ( factor2 * [-sin(E), sqrt_factor1 * cos(E) , 0d])

   tottime = tottime + timestep
endfor

if (numpoints eq 1) then begin
   sat_pos_in = reform(sat_pos_in)
   sat_vel_in = reform(sat_vel_in)
endif


End
```

```
;=====================================================================
;
; Description:
; ------------
; Procedure to transform a position vector from the inertial system to the
; Earth-fixed system.
;
; Inputs:
; -------
; pos_in        [xyz]  Position vector in inertial system
; starttime     [s]    Starting time of orbit observation
; timeduration  [s]    Duration of orbit observation
; timestep      [s]    Time between statevector outputs
;
; Outputs:
; --------
; pos_ef        [xyz]  Position vector in earth-fixed system
;
; Code maintainer:
; ----------------
;
; Richard Lord, 09-07-2004
;
; Reviewed by:
; ------------
; Shikoane G. Phaladi, April-2006
;
;=====================================================================


Pro transform_inertial_to_fixed, pos_in, pos_ef=pos_ef, starttime=starttime, $
                                 timeduration=timeduration, timestep=timestep


;-----Definition of global variables
@introduce_common_parameters.bat


;-----Get Earth parameters based on Earth-Model WGS84 and JGM3.
omega_earth  = constants.omega_earth    ; earth rotation rate [rad/s]


;-----Error checking
if ((not n_elements(starttime)) or (not n_elements(timeduration))) then begin
   message, 'Error: Missing starttime and timeduration!'
endif
if ((not n_elements(timestep)) and (timeduration ne 0d)) then begin
   message, 'Error: Missing timestep!'
endif

if (timeduration eq 0d) then begin
   numpoints  = 1L
endif else begin
   numpoints  = long(timeduration / timestep) + 1L
endelse

if (size(pos_in, /n_elements) / 3L ne numpoints) then begin
   message, 'Error: The input vector does not contain enough elements!'
endif
```

```
;-----Now advance until (starttime + timeduration), in timestep intervals
tottime = starttime


if (numpoints eq 1) then begin
   pos_ef   = reform(rotz(-omega_earth * tottime) ## pos_in)
endif else begin
   pos_ef = dblarr(numpoints, 3)
   for i=0L, (numpoints-1) do begin
      pos_ef[i, *] = rotz(-omega_earth * tottime) ## pos_in[i, *]
      tottime      = tottime + timestep
   endfor
endelse


End
```

```
;=====================================================================
;
; Description:
; ------------
; Procedure to calculate and view the subsatellite tracks of the
; GPS satellites on a world map.
;
; Code maintainer:
; ----------------
; Richard Lord, 08-07-2004
;
;=====================================================================


;-----Definition of global variables
@introduce_common_parameters.bat


;-----User input
starttime      = 0d
timeduration   = 24d * 3600d
timestep       = 500d
w              = 25d-2   ; approximate wavelength of the GPS receiver
n              = 1d      ; Fresnel number

;-----Compile all routines
compile_routines
resolve_routine, 'fresnel_ellipse'

;-----Initialise global parameters
define_gps_params


;-----Get constants and parameters
radeg       = constants.radeg
gm_earth    = constants.gm_earth
;num_gps_ids = n_elements(orbit_parameters.sat_name)
num_gps_ids = 1l
degrad      = constants.dtor

;----- Defining the receiver's position
;A = [-33.57*degrad, 18.28*degrad, 1100]     ; Table mountain
;B = [-34.00*degrad, 22.22*degrad, 1000]     ; George
C = [-33.02*degrad, 27.49*degrad , 6500000]     ; East London
;D = [-29.57*degrad, 30.56*degrad, 100000]     ; Durban

rec_latlon = C
convert_latlon_xyz, rec_latlon, xyz_pos=rec_pos

;-----Get orbit parameters
get_gps_orbit_parameters


;-----Initialise arrays
numpoints   = long(timeduration / timestep) + 1L
gps_latlonh = dblarr(num_gps_ids, numpoints, 3)
f_ellipse = dblarr(num_gps_ids, numpoints,3)
clock = dblarr(numpoints)
clock1 = dblarr(numpoints)
ref_latlonh = dblarr(num_gps_ids, numpoints, 3)
fresarray   = dblarr(num_gps_ids, numpoints)
```

```
f_angle   = dblarr(num_gps_ids, numpoints)


;-----For each GPS satellite
for i=0L, (num_gps_ids - 1L) do begin

   if (orbit_parameters.sat_name[i] ne '') then begin

;convert_latlon_xyz, rec_latlon, xyz_pos=rec_pos

      ;-----Get satellite statevectors in inertial system
      satellite_inertial_vectors, sat_pos_in, sat_vel_in, i, starttime=starttime, $
                                  timeduration=timeduration, timestep=timestep


      ;-----Get earth trace in inertial system
      off_nadir = 0d
      ellipsoidal_intersection, sat_pos_in, sat_vel_in, off_nadir, $
                                target_pos_in=swath_in


      ;-----Transform to earth-fixed system
      transform_inertial_to_fixed, swath_in, pos_ef=sat_pos_ef, starttime=starttime, $
                                   timeduration=timeduration, timestep=timestep



      ;-----Calculating the reflection point
      reflection_pos, sat_pos_ef, rec_pos, ref_xyz_pos =
ref_xyz_pos,starttime=starttime,$
                      timeduration=timeduration, timestep=timestep

      ;-----Calculating the Fresnel zones
      fresnel,w, n, rec_pos, sat_pos_ef, ref_xyz_pos, f_radius =
temp,starttime=starttime,$
                      timeduration=timeduration, timestep=timestep, timer = timer

      ;----calculating the reflection angle
      reflection_angle, sat_pos_ef, rec_pos, ref_angle=final_angle,
starttime=starttime, $
                     timeduration=timeduration, timestep=timestep,timer1 = timer1

      ;------Calculating the fresnel ellipse
      fresnel_ellipse,final_angle, n, w, sat_pos_ef, rec_pos, ellipse_par = final,
starttime=starttime,$
                      timeduration=timeduration, timestep=timestep



      ;-----calculating the semimajor of the fresnel ellipse
      ;fresnel_smajor, temp, final_angle, sem = smajor, starttime=starttime, $
       ;               timeduration=timeduration, timestep=timestep



      ;-----Conversion from Cartesion to Spherical coordinate system
      convert_xyz_latlon, sat_pos_ef, latlonh=sat_latlonh
```

```
      ;-----Conversion from Cartesion to Spherical coordinate system
      convert_xyz_latlon, ref_xyz_pos, latlonh=ref_pos


      ;-----Put into large array
      gps_latlonh[i, *, *] = sat_latlonh
      f_ellipse[i,*] = final
      ;clock[i] = timer
     ; clock1[i] = timer1
      ;print, ellipse_par
     ;ref_latlonh[i, *, *] = ref_pos
      ;fresarray[i,*] = temp
      ;f_angle[i,*] = final_angle




   endif

;print,f_ellipse



endfor



;print, f_angle[*,*]

;print, f_ellipse

;print,gps_latlonh[*, *, 2]

;SET_PLOT, 'PS'
;DEVICE, /ENCAPSULATED, FILENAME = 'EL-1km.ps',/COLOR, BITS=8
;device,retain = 2
;plot,gps_latlonh[*,*,1]*radeg,f_ellipse[*],xrange = [20,40]
;plot,  fresarray
;plot,f_angle
;plot, fresarray[*,*,*]
;wait,2
;oplot,gps_latlonh
;plot, clock,f_ellipse[*];,LINESTYLE = 2, TITLE = 'East London GPS receiver at 1-km
height', $
;XTITLE = 'iterations', YTITLE='Fresnel radius [m]'
;xrange = [0,50]
;oplot, gps_latlonh[*,*,1]*radeg,fresarray[*]




;-----Plot GPS orbits on world map
;plot_map, /grid, /cylindrical
```

```
;stepsize = 2
;for k=0L, (numpoints - 1L), stepsize do begin


;   for i=0L, (num_gps_ids - 1L) do begin



;      if (orbit_parameters.sat_name[i] ne '') then begin


;         plot_map, lon_arr=[rec_latlon[1]],$
;                   lat_arr=[rec_latlon[0]],$
;                   /grid, /cylindrical,/keepoldplot,/color,/psym


;        plot_map, lon_arr=[ref_latlonh[i, k:k+stepsize, 1]], $
;                   lat_arr=[ref_latlonh[i, k:k+stepsize, 0]], $
;                   /grid, /cylindrical,/keepoldplot,/psym



;         plot_map, lon_arr=[gps_latlonh[i, k:k+stepsize, 1]], $
;                    lat_arr=[gps_latlonh[i, k:k+stepsize, 0]], $
;                   /grid, /cylindrical,/keepoldplot,/psym



;      endif
;   endfor
;  wait, 2
;   plot_map, /grid, /cylindrical, /keepoldplot, /clean
;endfor




End
```