A SATELLITE FOOTPRINT VISUALISATION TOOL

Master-Thesis

VIS SAT

by Patrick Daum
Department of Communication Systems

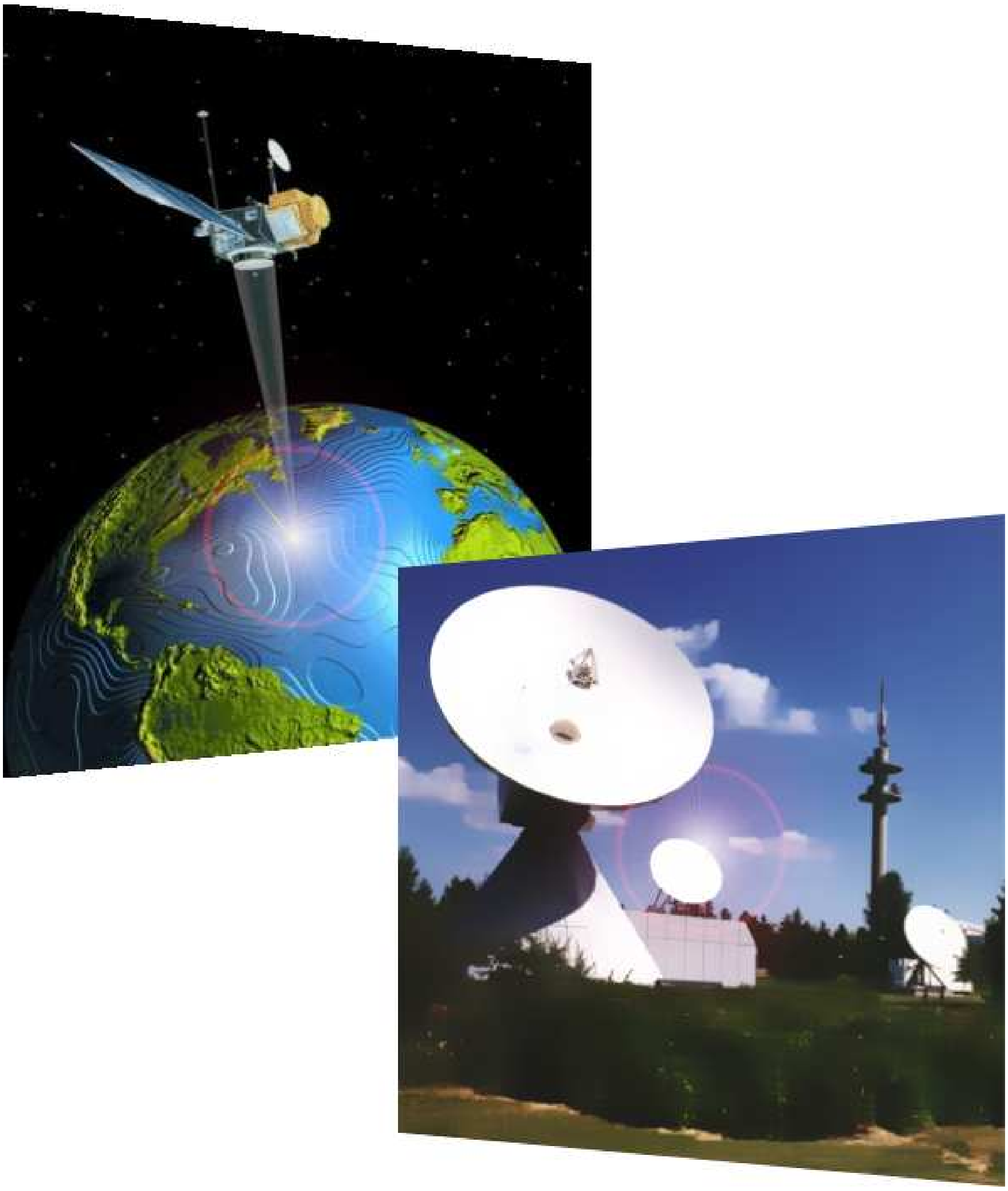
September 2005

Dissertation submitted in partial fulfillment of the
MSc in Satellite Communications and Space Environment

Supervisors: Dr. J.A. Wild and Dr. A.J. Kavanagh

Lancaster University
attn.: P.Daum
44581 Castrop-Rauxel
GERMANY
Phone: (+49) 2367-98044
EMail: `pdaum@mail2daum.de`
www.vissat.net.ms

MATLAB® is a registered trademark of
The MathWorks Inc.
For production information, please contact:
The MathWorks Inc.
3 Apple Hill Drive
Natick, MA 01760-2098, USA
Phone: (508) 647-7000
EMail: `info@mathworks.com`
www.mathworks.com



© ENSO

Abstract

This thesis concerns the development of a numerical software tool for propagating and visualising orbits of artificial bodies orbiting the Earth and the calculation of magnetic conjunctions between these objects and the ground. The developed toolbox **VIS SAT** enables the users to visualise the trajectories of satellites under different parameters and to estimate the visibility of a satellite to maintain efficient satellite ground communication links. The trajectory predictions enables the user to visualise the magnetic conjunctions for "space weather" studies by plotting the northern and southern footprint of the magnetic field line.

The satellite's motion results from the analysis of all forces which have an effect on the trajectory of the satellite. The resultant of all forces acting on a satellite is proportional to its acceleration, so it is possible to describe the trajectory of a satellite in the central force field it is acting in. In this case the main acting force is the gravitational force. This force is very variable in the near Earth space due to the effect of the Earth's flattening and is therefore very variably marked; it could be characterised by the zonal harmonics. The equations to describe the motion of a satellite arises from Kepler's empirically laws and Newton's mathematical description afterwards. The major focus here lies on Newton's law of universal gravitation and Newtons second law of motion. These laws leads to a description of the orbital motion by using differential equations, which are only possible to solve analytically when an ideal gravitational force field could be assumed but the Earth is known to neither have an even mass distribution nor be a perfect sphere. Therefore in order to get a precise description of the orbital motion the physical laws have to be expanded. This is done by using the Simplified General Perturbation Model (SGP4/SDP4) which was developed by the North American Aerospace Defense Command (NORAD) and implementing the methods, described in this model, in MATLAB®. The determination of the magnetic conjunction is performed by Tsyganenko's magnetic field model (Tsy'03). By means of this model and the numerical representation in MATLAB® the magnetic connection between the satellite and the ground station can be determined.

All these models are summarised in three basic functions of **VIS SAT** in order to facilitate a fast numerical calculation of the given equations of motion and field descriptions, so it is possible to receive accurate results which agree with the theory in the context of computed precision.

The toolbox **VIS SAT**, the online documentation and all functions can be downloaded from the following website

HTTP://WWW.VISSAT.NET.MS

Also there is an attached CD-ROM with all functions and models used in the thesis.

Contents

1 Introduction 1

2 Satellite Orbit Dynamics and Kinematics 3

2.1 Coordinate Systems 3

2.1.1 Earth Centered Inertial - ECI 4

2.1.2 Earth Centered Earth Fixed - ECEF 4

2.1.3 Topocentric Horizon Coordinate System 4

2.1.4 Orbit Coordinate System 5

2.2 Newton's laws 5

2.2.1 General Considerations 5

2.2.2 Two body Problem 6

2.3 Kepler's laws 10

2.3.1 Determination of the orbit shape 10

2.3.2 Conservation of Energy 10

2.3.3 Time derivation of the orbit 12

2.3.4 Keplerian Elements 15

2.4 NORAD Orbit Propagation 19

2.4.1 NORAD TLE 19

2.4.2 Simplified General Perturbation Model 21

2.4.2.1 Perturbations due to Geopotential 21

2.4.2.2 Perturbations due to the Sun and the Moon 22

2.4.3 Simplified General Perturbation Model Version 4 22

2.4.3.1 Perturbations due to Atmospheric Drag 23

2.4.3.2 Perturbations due to Solar Radiation 23

2.5 Satellite Orbits 23

2.5.1 Synchronous Orbits 24

2.5.1.1 Earth Synchronous 24

2.5.1.2 Sun Synchronous 24

2.5.2 Polar Orbits 25

2.5.3 Recurrent Orbit 25

2.6 Summary 25

3 Magnetic Coupling Model 26

3.1 Earth Magnetic Field 26

3.2 Tsyganenko's Magnetic Field Model 28

3.3 Field Line Tracing 30

3.4 Summary 32

4 Program/Toolbox (VIS SAT) 34

4.1 Structure and Implementation 34

 4.1.1 Read NORAD TLE 36

 4.1.2 SGP4 and SDP4 implementation (`MEX` Function `orbit`) 37

 4.1.3 Satellite Position Footprint (`SatPOS`) 40

 4.1.4 Ground Station Tracking (`SatTRACK`) 44

 4.1.5 Magnetic Footprint (`MagPOS`) 47

4.2 Graphical User Interface 49

 4.2.1 Operation and functionality 51

  4.2.1.1 Satellite Input Data 51

  4.2.1.2 Time Settings 52

  4.2.1.3 Magnetospheric Parameters 53

  4.2.1.4 Simulation and View 53

4.3 Plot Functions 54

 4.3.1 Plot on 2D Map (`plot2dmap`) 54

 4.3.2 Plot on Blue Marble Maps (`plotonmarble`) 54

 4.3.3 Plot Skymap (`skymap`) 55

4.4 Summary 56

5 Conclusion and Future Work 58

References 61

A Equations, Tables, Explanations 65

A.1 Geopotential 65

A.2 Earth Atmosphere 67

A.3 Time-Variant Kepler Elements 67

A.4 Detailed SGP Model 69

A.4.1 SGP Model 69
A.4.2 SGP4 Model 70
A.4.3 SDP4 Model 74
A.4.4 Boundaries of the NORAD models 75
A.5 Additional Accelerations 77
A.6 Earth's Magnetic Field Additions 77
A.6.1 Quasi Schmidt normalised 77

B MATLAB **79**

B.1 Explanation: MATLAB® /`MEX` 79
B.2 `norad2kep.m` 79
B.3 `orbit.m` 81
B.4 `SatPOS.m` 99
B.5 `SatTRACK.m` 100
B.6 `MagPOS.m` 102
B.7 Validation 104

C `GEOPACK` **104**

List of Figures

2.1 Sputnik 1 3
2.2 Earth Centered Inertial Coordinate System 4
2.3 Topocentric Horizon Coordinate System 4
2.4 Orbit Coordinate System 5
2.5 Two Body Problem 6
2.6 Plane Motion 8
2.7 Conic Sections 9
2.8 Keplerian Orbit 11
2.9 Motional Integral, Angular Momentum 12
2.10 Illustration of the Eccentric Anomaly 13
2.11 Keplerian Elements 15
2.12 Keplerian Orbit with Perturbations 21

2.13 Earth Synchronous Orbits . . . 24
2.14 Recurrent Orbit . . . 25
3.1 Three Dimensional Structure of the Earth's Magnetic Field . . . 27
3.2 Earth's Dipole Field under different outer Conditions . . . 28
3.3 Magnetosphere . . . 29
3.4 Tsyganenko's Geomagnetic Field Model . . . 31
3.5 Simple Shapes of the Earth's magnetic field . . . 33
4.1 ECI to Geodetic Conversion . . . 41
4.2 Geodetic Position Footprints . . . 42
4.3 Three Dimensional Simulated Trajectory . . . 43
4.4 Skymap . . . 44
4.5 Ground Station . . . 45
4.6 Skymap plots to track Satellites . . . 46
4.7 Magnetic Conjunction Footprint . . . 49
4.8 Graphical User Interface . . . 50
4.9 Satellite Data . . . 51
4.10 Osculating Element Perturbation . . . 52
4.11 Time Setting . . . 53
4.12 Magnetospheric Setting . . . 53
4.13 Geodetic Position Footprint on Marble . . . 55
5.1 Tracking Device . . . 60
A.1 Geopotential . . . 66
A.2 Nutation . . . 76
A.3 Accelerations on Earth orbiting Satellites . . . 77

List of Tables

2.1 Keplerian Elements . . . 15
4.1 Array structure of TLE . . . 37
4.2 Assignment Structure between MATLAB® and C/C++ . . . 39
B.1 GODDARD SSC Locator Values . . . 104

List of mathematical abbreviations

a	Semimajor Axis
A	Cross Section Area
$\vec{A}$	Integration Constant for the Orbit plane
b	Semiminor Axis
B^*	Ballistic Coefficient
c_w	Drag Coefficient
$\vec{C}$	Plane Vector for the Orbit
e	Eccentricity
e_x	Unity Vector in x Direction of the given Reference Frame
e_y	Unity Vector in y Direction of the given Reference Frame
e_z	Unity Vector in z Direction of the given Reference Frame
E	Eccentric Anomaly
G	Gravitational Constant
h	Energy Constant for Central Force Fields
i	Inclination
k	Number of Revolutions per Day
L	Angular Momentum
m_s	Mass of the Satellite
M	Mean Anomaly
M_E	Mass of the Earth
n	Mean Angular Velocity
ν	True Anomaly
$\dot{\nu}$	Angular Velocity of the Satellite
ω	Argument of Perigee
Ω	Right Ascension of Ascending Node (RAAN)
p	Ellipse Parameter
q_0	Density of the Atmosphere
$\vec{r}$	Direction Vector
$\dot{\vec{r}}$	Velocity
$\ddot{\vec{r}}$	Acceleration
$\ddot{\vec{r}}_{\text{Drag}}$	Acceleration due to Atmospheric Drag
$\ddot{\vec{r}}_{\text{solar}}$	Acceleration due to Solar Perturbations
t_0	Epoch Time or Time of Perigee
T	Period of the Elliptical Orbit

List of abbreviations

AU	Astronomical Unit
CBAT	Central Bureau for Astronomical Telegrams
CoM	Center of Mass
DGRF	Definitive Geomagnetic Reference Field
DST	Disturbance Storm Time Index
ECEF	Earth Centered Earth Fixed Coordinate System
ECI	Earth Centered Inertial Coordinate System
ESA	European Space Agency
GHAA	Greenwich Hour Angle Aries
GMST	Greenwich Mean Sidereal Time
GSM	Geocentric Solar Megnatospheric System
IAGA	International Association of Geomagnetism and Aeronomy
IAU	International Astronomical Union
ICQ	International Comet Quarterly
IGRF	International Geomagnetic Reference Field
IMF	Interplanetary Magnetic Field
JD	Julian Date
MHD	Magneto Hydrodynamics
MPC	Minor Planet Center
NORAD	North American Aerospace Defense Command
RAAN	Right Ascension of Ascending Node
SATCAT	Satellite Catalog, database containing the location of all satellites
SDP4	Simplified General Deep Space Perturbation Model (Version 4)
SGP	Simplified General Perturbation Model
SGP4	Simplified General Perturbation Model (Version 4)
SSP	Subsatellite Point
TDB	Barycentric Dynamical Time
TDT	Terrestrial Dynamical Time
TLE	Two Line Elements
TSY'03	Tsyganenko '03 Magnetic Field Model
UT/UTC	Universal Time
WGS	World Geodetic System

Glossary

Altitude

The angular distance from the observer's horizon, usually taken to be the horizon that is unobstructed by natural or artificial features (such as mountains or buildings), measured directly up from the horizon towards the zenith.

Angular Momentum

Angular momentum measures an object's tendency to continue to spin. The angular momentum of a particle of mass m with respect to a chosen origin is given by the vector product between the direction vector and the impulse vector.

Apogee

The point where (and when) an object's orbiting a huge massive body is farthest to this body.

Arc minutes

There are 60 minutes (denoted as 60') of arc in 1 degree. In the sky, with an unobstructed horizon (as on the ocean), one can see about 180 degrees of sky at once, and there are 90 degrees from the true horizon to the zenith. The full moon is about 30' (30 arc minutes) across, or half a degree. There are 60 seconds (denoted 60") of arc in one minute of arc.

Azimuth

Angular distance measured clockwise around the observer's horizon in units of degrees; north to be 0 degrees, east to be 90 degrees, south to be 180 degrees, and west to be 270 degrees.

Besselian year

A quantity introduced by F.W. Bessel (*1784-†1846) in the nineteenth century that has been used into the twentieth century. Bessel introduced a system whereby it would be convenient to identify any instant of time by giving the year and the decimal fraction of the year to a few places, but the starting times of the year was not convenient for dynamical studies that utilise Julian dates (see definition for Julian date), differing by 0.5 day, and the Besselian year varies slowly. The recent change to Julian year usage in dynamical astronomy (and the J2000.0 equinox) took effect in solar-system ephemeris of the Minor Planet Center and Central Bureau for Astronomical Telegrams on Jan. 1, 1992 (See Julian year).

Celestial sphere

An imaginary sphere of great (or infinite) radius that is centered on the Earth and is used for practical purposes in astronomical observing. Since stars are very distant from the Earth, they make up a background that is essentially unchanging from year to year; over a period of years, the closer stars will move very slightly and factors such as precession cause a change in the appearance of the stars in the skies over many years. Therefore it is necessary to create a map grid on the celestial sphere for identifying, referring to, and locating objects in the sky; some of these map grids include equatorial coordinates (right ascension and declination), ecliptic coordinates (ecliptic longitude and latitude), and galactic coordinates (galactic longitude and latitude) - which refer to the Earth's rotation, the Earth's revolution about the Sun, and the Milky Way galaxy's plane, respectively.

Degree

A unit used in the measurement of angles. Due to ancient Babylonian mathematics, a circle is still divided into 360 even units of arc and each of these units are called one degree. The entire sky, therefore, spans 360 degrees. Up to about 180 degrees of the sky is visible from any given point on Earth with an unobstructed horizon (as measured from, east to west, or north to south). The degree is used to make measurements of distance, or position (as with declination) in astronomy. In turn, a degree is composed of 60 minutes of arc, and also of 360 seconds of arc.

Drag Term (`BSTAR`)

The drag term `BSTAR` is an SGP4 type drag coefficient. In aerodynamic theory every object has a ballistic coefficient B that is the product of its coefficient of drag c_w and its cross sectional area A divided by its mass m. The ballistic coefficient represents how susceptible an object is to drag - the higher the number, the more susceptible. `BSTAR` is an adjusted value of B using the reference value of atmospheric density q_0 in the SGP models.

Ecliptic

The apparent path of the Sun against the sky background (celestial sphere); formally, the mean plane of the Earth's orbit about the Sun.

Elongation

Angular distance of a celestial object from the Sun in the sky. In standard ephemeris, this is usually denoted by the greek letter epsilon ϵ (or by the abbreviation "Elong"). A celestial (usually solar-system) object's "phase angle" is the elongation of the Earth from the Sun, as would be seen by an observer on that third celestial object.

Epoch

Time of the Keplerian elements, the Keplerian elements describe the orbit and the epoch times describes the time in which the satellite was at the exact position in respect to the Keplerian elements which are stated in the TLE format. The Epoch time standard in the NORAD TLE data is `yyddd.ffffffff`, where the `yy` is equal to the last two digits of the year, so 00 <`yy`$\leq$ 56 corresponds to 2000-2056 above 56 corresponds to the years 1957-1999. The `ddd` is the number of the day in this year, the `.ffffffff` corresponds to the fraction of this day.

Equinox

Either of the two points (vernal, autumnal) on the celestial sphere where the ecliptic intersects the celestial equator. Due to precession, this point moves over time, so positions of stars in catalogues and on atlases are usually referred to a "mean equator and equinox" of a specified standard epoch. For the propagation purposes of the positions of objects the positions are always given for "equinox J2000.0", meaning that the reference system is that at the beginning of the year 2000; prior to 1992. The B and J preceding the equinox years indicate "Besselian" and "Julian", respectively. See separate definitions for Besselian year and Julian year. The differences in an object's position when given in equinoxes 1950.0 and 2000.0 amounts to several arc minutes.

Heliocentric

Referring to the Sun. A heliocentric orbit is one based on the Sun as one of the two foci of the (elliptical) orbit (or as the center of a circular orbit); a heliocentric magnitude is the brightness of an object as would be seen from a heliocentric distance of 1 astronomical unit (which means a distance of 1 AU from the Sun).

Julian date (JD)

The interval of time in days (and fraction of a day) since Greenwich noon on Jan. 1, 4713 BC. The JD is always half a day off from Universal Time, because the current definition of JD was introduced when the astronomical day was defined to start at noon (prior to 1925) instead of midnight. Thus, 1995 Oct. 10.0 UT = JD 2450000.5.

Julian year

Exactly 365.25 days, in which a century (100 years) is exactly 36525 days and in which 1900.0 corresponds exactly to 1900 January 0.5 (from the Julian-date system, which is half a day different from civil time or UT). The standard epoch J2000.0, now used for new star-position catalogues and in solar-system-orbital calculations, means 2000 Jan. 1.5 Barycentric Dynamical Time (TDB) = Julian Date 2451545.0 TDB. When this dynamical, artificial "Julian year" is employed, a letter "J" prefixes the year.

Orbit

The path of one object about another (used here for an object orbiting a massive body in the central force field of its gravity).

Orbital elements

Parameters (numbers) that determine an object's location and motion in its orbit about another object. In the case of solar-system objects such as comets and planets, one must ultimately account for perturbing gravitational effects of numerous other planets in the solar system (not merely the Sun), and when such account is made, the "osculating elements" (which are always changing with time and which therefore must have a stated epoch of validity) are determined. Six elements are usually used to determine uniquely the orbit, with a seventh element (the epoch, or time, for which the elements are valid) added when planetary perturbations are allowed for; initial ("preliminary") orbit determinations shortly after the discovery of a new comet or minor planet (when very few observations are available) are usually "two-body determinations", meaning that only the object and the massive body are taken into account) work with only the following six orbital elements: time of perihelion passage (sometimes taken instead as an angular measure called "mean anomaly"); perihelion distance, eccentricity of the orbit; and three angles (for which the mean equinox must be specified) - the argument of perihelion, the longitude of the ascending node, and the inclination of the orbit; these orbit elements are called the Keplerian elements.

Perigee

The point where (and when) an object's orbit about a massive body is closest to this body.

Perturbations

Gravitational influences ("tugging" and "pulling") of one astronomical body on another. Comets are strongly perturbed by the gravitational forces of the major planets, particularly by the largest planet, Jupiter. These perturbations must be allowed for in orbit computations, and they lead to what are known as "osculating elements" which means that the orbital element numbers change from day to day and month to month due to continued perturbations by the major planets, so that an epoch is necessarily stated to denote the particular date that the elements are valid. Also different forces have to be considered as cause of perturbations like the atmospheric drag, solar radiation and the geopotential.

Precession

A slow but relatively uniform motion of the Earth's rotational axis that causes changes in the coordinate systems used for mapping the sky. The Earth's axis of rotation does not always point in the same direction, due to gravitational tugs by the Sun and Moon (known as lunisolar precession) and by the major planets (known as planetary precession).

Right ascension

One element of the astronomical coordinate system on the sky, which can be though of as longitude on the Earth projected onto the sky. Right ascension is usually measured eastward in hours, minutes, and seconds of time from the vernal equinox. There are 24 hours of right ascension, though the 24-hour line is always taken as 0 hours. More rarely, one sometimes sees right ascension in degrees, in which case there are 360 degrees of right ascension to make a complete circuit of the sky.

Sidereal Time

The sidereal time is the time measured by the apparent diurnal motion of the vernal equinox. Sidereal time is defined as the hour angle of the vernal equinox. When the meridian of the vernal equinox is directly overhead, local sidereal time is 00:00. Greenwich sidereal time is the hour angle of the vernal equinox at the prime meridian at Greenwich. Greenwich sidereal time and UT1 differ from each other by a constant rate 1.00273790935.

Simplified General Perturbation

The Simplified General Perturbation model is an advanced model of a Keplerian orbit model. Ideally a satellite - once put into orbit - would precisely repeat its orbit forever. In reality the satellite will slowly lose energy due to gravity forces and atmospheric drag which it encounters. The Simplified General Perturbation model takes this into account (further details see [20]).

Subsatellite Point

Point where a straight line drawn from a satellite to the center of the Earth (position vector $\vec{r}$) intersects the Earth's surface.

Terrestrial Dynamical Time (TDT or TT)

Time scale used in orbital computations; TDT is tied to atomic clocks (International Atomic Time, TAI), whereas Universal Time is tied to observations. Prior to 1992, Ephemeris Time (ET) was used in publications of the ICQ/CBAT/MPC; since then, TT has been used. The difference between TDT and UTC in 1994 was 60 seconds (i.e., UT + 60 seconds = TDT).

Universal Time (UT, or UTC)

A measure of time used by astronomers; UT conforms (within a close approximation) to the mean daily (apparent) motion of the Sun. UT is determined from observations of the diurnal motions of the stars for an observer on the Earth. UT is usually used for astronomical observations, while Terrestrial Dynamical Time (TDT, or simply TT) is used in orbital and ephemeris computations that involve geocentric computations. Coordinated Universal Time (UTC) is that used for broadcast time signals (available via shortwave radio, for example), and it is within a second of UT.

Vernal equinox

The point on the celestial sphere where the Sun crosses the celestial equator moving northward, which corresponds to the beginning of spring in the northern hemisphere and the beginning of autumn in the southern hemisphere (in the third week of March). This point corresponds to zero hours of right ascension.

1 Introduction

Challenging or frustrating, fascinating or confusing - no matter how they are described, space satellites certainly have added a new dimension to Communication Systems and the modern society. The up and down link connections between the ground and the *"cyberelectric"* cocoon that envelopes the entire planet are essential for our modern society. An accurate knowledge of the position of a satellite over the ground is necessary to establish a reliable link between the ground station and the spacecraft. The basis of the satellite's motion are the laws from celestial mechanics which are based on the simpler physical laws of motion. The visualisation of such motions have become indispensable in modern science, the visualisation is a tool to show complex mathematical issues in a simpler way. So it is possible to use these visualisations to get a deeper insight in procedures and to simulate different behaviours under special circumstances.

One of the basis of these visualisations is the numeric which acts as a link between an analytical equation which characterise a scientific problem and the calculation on a computer. These methods describe therefore the reality by numbers. These numbers are feasible for computer or for the mathematic but unmanageable for a human being. Therefore it is necessary to preprocess the data. For this purpose technical software packages like MATLAB® (for ***MAT****rix* ***LAB****oratory*) by `MathWorks` are in use to solve numerical problems and to visualise the huge amount of data in an understandable way. Here it is essential to find the right numeric method for a given physical problem because the methods differ in accuracy, stability, efficiency and velocity. This is one major task of this thesis, to solve the differential equations of motion in a fast and stable way by using pre existing numerical functions and models. A complex physical problem could be solved by combining different methods in a sequential or interlinked way but a little change in the physical circumstances could result in a total different way of solving the problem.

The main field of application for this thesis is to visualise the motion of satellites in a near Earth environment and to develop a toolbox in MATLAB® which provides functions for tracking the satellites over the ground. In order to do so known equations from celestial mechanics are used and the virtual trajectory of a satellite is simulated in MATLAB® . Orbital mechanics, as applied to artificial Earth satellites, is based on celestial mechanics, a branch of ordinary mechanics which started with Newton and Kepler. A quantitative analysis of the motion is based on Newton's law of universal gravitation and Newton's second law of motion. The fundamental properties of the orbits are summarised in Kepler's three laws of planetary motions which are also applicable for artificial bodies in space orbiting a huge mass body. These laws contains the differential equations which have to be considered and solved in order to simulate the trajectories of the satellites orbiting the Earth. The first part of this thesis shows the theory and equations used to solve the mathematical two body problem for artificial satellites. The second part shows how these equations can be applied in a software tool. In addition to the motional visualisation this tool also provides a function to estimate magnetic footprints. The satellites are exposed to the Earth magnetic field in their orbits, here the cross section of these exposed areas with the magnetic field lines are of interest. With pre existing models it is possible to visualise footprints of these field lines on the ground during the orbit period.

The program **Vis Sat** forms the end of this thesis and the functionality description and program documentation is the last part of this thesis. This program/toolbox is capable of calculating the trajectory of a satellite by using a graphical user interface. The stand alone functions contained in the toolbox could also be used in a command line way in the MATLAB® environment for a future development. **Vis Sat** is a scientific tool to study the satellite/ground station conjunction and also to study the magnetic conjunction. The GUI of **Vis Sat** acts as a link between the parameterised physical equations and a descriptive visualisation of the motions.

2 Satellite Orbit Dynamics and Kinematics

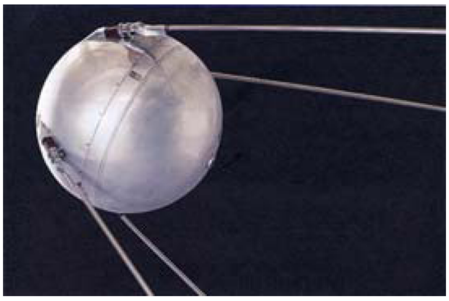


Figure 2.1: Sputnik 1, the first artificial satellite (source: [28])

Since the first laws of celestial mechanics were developed by Johannes Kepler (*1571 - †1630) and Sir Isaac Newton (*1642 - †1727) mankind has been fascinated by space and the physics behind the processes in space. Since this time it was a dream to create artificial satellites and to bring them in an orbit around the planet but until the launch of *Sputnik 1* this was just theoretical possible. The ascent of *Sputnik 1* (see fig. 2.1), on the 4th October 1957 ushered in the space age [32] *"... the satellite that inspired generations"*. From this day on a lot more satellites have been launched and today they form a complex *"cyberelectric"* cocoon (first mentioned in [1], Laboratory for Atmospheric and Space Physics) that envelopes the entire planet. Modern satellites perform various task from navigation, broadcasting, imaging, scientific to communication and even spy purposes. It is not possible to imagine life in a modern society without these satellites. Nowadays communication around the globe is possible by using various up and down link connections, scientific satellites explore the near Earth environment and even deep space. But all satellites, artificial or not, obey the basic physical laws of celestial mechanics. With these basic laws it is possible to pre calculate the position of a body in orbit.

At the beginning these physical laws should be introduced to form the theoretical background. The first part deals with the mathematics of satellite orbits, starting with the basics of Newton's laws and the original Keplerian orbit model, the classic orbital elements and the calculations to describe the orbit. Derived from this basic model, the NASA and the US Space Command derived the Simplified General Perturbation models (SGP). These models are more advanced than the Keplerian model, in that they take into account forces beyond the Earth's gravitation (exact description of the additional forces see section A.1, A.2 and A.5), which cause perturbations in the actual orbit. The primary models, SGP and SGP4/SDP4 will be described in detail. This chapter will also give a description of the NORAD two-line element data (TLE), its relation to the Keplerian elements and how they will be used in the calculations. In order to get a deeper insight of the flight mechanics it is necessary to have a closer look at the forces which acts on a satellite in orbit and how they can influence the trajectory. In order to build up such theoretical models it is initially necessary to have a closer look at the frames of references which have to be used. The last part of this thesis deals with the implementation in MATLAB® and the graphical representations.

2.1 Coordinate Systems

In order to describe the dynamics and kinematics of satellites it is necessary to declare a (quasi-) inertial frame of reference (cf. [16, 17]). This means it is a frame in which the observers move without the influence of any accelerating or decelerating force. In case of the Earth satellite system the inertial frame is a space-time coordinate system that neither rotates nor accelerates. With this inertial frame it is possible to compute the satellite's trajectory by using vectorial transformations. With such frames of reference it is possible to achieve solvable equations of motion.

2.1.1 Earth Centered Inertial - ECI

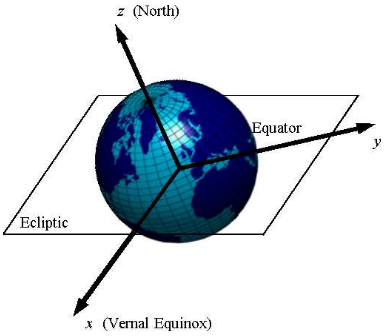


Figure 2.2: Earth Centered Inertial coord. system

The Earth Centered Inertial (ECI) coordinate system has its origin in the Earth's center. The x-axis is in the equatorial plane and points to the vernal equinox, the z-axis points to the north pole of the Earth and is parallel to the rotation axis of the Earth. The y-axis supplements the two other axes to a right-handed coordinate system. The ECI system is used as a reference system for the attitude description. Most of the attitude quaternions are calculated with respect to the ECI frame, e.g. the current nadir quaternion or the current target pointing quaternion.

2.1.2 Earth Centered Earth Fixed - ECEF

The origin of the Earth Centered Earth Fixed (ECEF) coordinate system is also located at the geocenter. The x-axis is in the equatorial plane and points in the direction of the Greenwich meridian. The z-axis points to the north pole and is parallel to the rotation axis of the Earth. The y-axis supplements the two other axes to a right-handed coordinate system. The ECEF system is important for the calculation of the sight times, the ground track of the satellite or the current target position.

2.1.3 Topocentric Horizon Coordinate System

The Topocentric Horizon Coordinate System is important for visibility considerations of the satellite. The origin is located in a certain position on Earth, e.g. target position or ground station position, thus the system moves with the Earth's rotation. Unfortunately two slightly different definitions exist for this coordinate system. Using the first, the z-axis is defined by the nadir direction at the given position which is the geocentric radius vector. Thus, the z-axis is the normal vector to a plane that defines the horizon on a sphere. In this plane the x-axis points south and the y-axis points east. The second definition defines the z-axis as the normal vector to the local horizon plane. Thus, the z-axis is the geodetic radius vector. The x- and y-axis are defined as aforementioned. This coordinate system is particularly useful for the calculation of sight times and look angles.

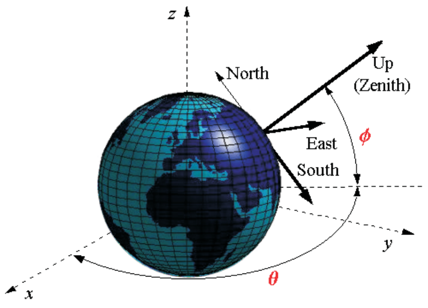


Figure 2.3: Topocentric Horizon coord. system

2.1.4 Orbit Coordinate System

The origin of the Orbit Coordinate System is in the satellite's CoM. The z-axis points in the direction of the nadir point, i.e. in the direction of the negative position vector of the satellite in the ECI system. The y-axis is perpendicular to the orbital plane and the x-axis point supplements the system to a right-handed orthogonal system and points in the flight direction (note that the x-axis is not necessarily parallel to the velocity vector for non-circular orbits).

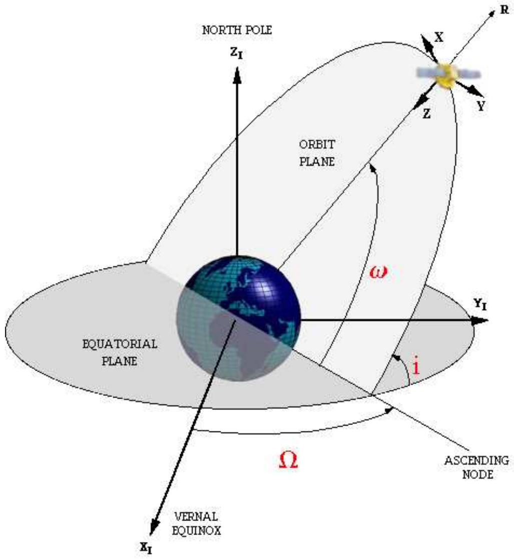


Figure 2.4: Orbit coord. system

2.2 Newton's laws

All motions are summarised in the physical field of mechanics, a branch of this field is celestial mechanics which makes use of the frames of reference stated above. The first mathematical exact laws in this field were developed by Isaac Newton, Newton's laws of motion, together with his law of universal gravitation and the mathematical techniques of calculus, provided for the first time an unified quantitative explanation for a wide range of physical phenomena such as: the motion of spinning bodies, motion of bodies in fluids; projectiles; motion on an inclined plane; motion of a pendulum; the tides; the orbits of the Moon and the planets. These basic laws are still valid and they are essential to describe the motion of satellites in orbit.

2.2.1 General Considerations

The fundamental laws of physics upon which the theory of orbital mechanics is based are Newton's law of universal gravitation and Newton's law of motion. The law of gravitation states that the gravitational force attraction between two bodies varies as the product of their masses M and m (here M_E and m_s) and inversely as the square of the distance $|r|$ between them and is direct along a line connecting their centers. Thus

$$\vec{F} = -\frac{GMm}{|r|^2}\frac{\vec{r}}{|r|} \tag{2.1}$$

where $G = 6.67 \cdot 10^{-11}$ $\mathrm{Nm}^2/\mathrm{kg}^2$ is the universal gravitational constant[1]. The second law of motion states that the acceleration of a body is proportional to the force acting on it and is inversely proportional to its mass, this leads to

$$\vec{F} = m\ddot{\vec{r}} \tag{2.2}$$

where $\ddot{\vec{r}}$ is the acceleration. The vector $\vec{r}$ is from M to m and the force is on m. These equations can also include terms for non gravitational disturbances such as atmospheric drag or they could also include additional gravitational perturbations which results from the non spherical shape of the Earth. But these additional forces could be very complex and a closed form solution for these kind of equations is impossible. That is why a lot of numerical computer models are available to include such disturbances (see appendix A.1, A.2, A.5 and [33]). In order to keep the theoretical considerations as simple as possible these additional disturbances could be considered as a time depending constant (see details in SGP model). However for artificial Earth satellites many important results can be found by considering the two bodies (cf. fig. 2.5) in a special reference frame. The reference frame is X, Y, Z as one in which Newton's law applies.

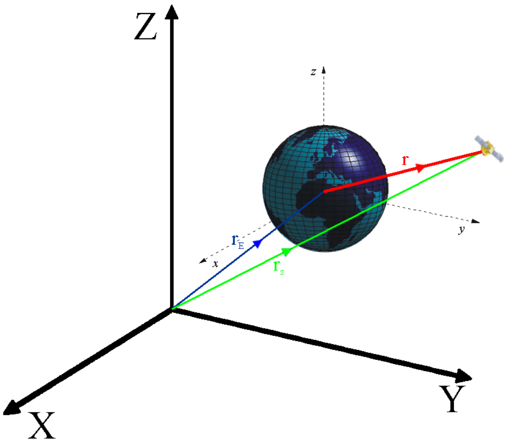


Figure 2.5: Geometry of the two body problem

2.2.2 Two body Problem

In order to analyse the two body problem the two bodies (here the Earth and the satellite) are considered one with the mass M (Earth) and the mass m (satellite).

---

[1] first measurement by Cavendish 1798 with a torsions balance

Then with equation (2.1) and equation (2.2) it can be written

$$\begin{aligned} \vec{F}_m &= m\ddot{\vec{r}}_m = -\frac{GMm}{|r|^2}\,\frac{\vec{r}}{|r|} \qquad (2.3) \\ \vec{F}_M &= m\ddot{\vec{r}}_M = +\frac{GMm}{|r|^2}\,\frac{\vec{r}}{|r|} \qquad (2.4) \\ & \qquad (2.5) \end{aligned}$$

where $\vec{r} = \vec{r}_m - \vec{r}_M$. Subtraction yields to

$$\ddot{\vec{r}} = -\frac{G(M + m_{corr})}{|r|^3}\,\vec{r}\,. \tag{2.6}$$

This is the basic vector differential equation for the two body problem which specifies the acceleration of the body of mass $m_{corr} = Mm/(M+m)$ with respect to the body of mass M. This problem can then be solved using different frames of reference (see in section 2.1). For the next considerations it can be assumed that the mass of a satellite is $m \ll M$ and $G(M+m) \approx GM$. Equation (2.6) then becomes

$$\ddot{\vec{r}} = -\frac{GM}{|r|^2}\,\frac{\vec{r}}{|r|} \qquad \text{(general equation of motion)}\,. \tag{2.7}$$

All models for satellite propagation are based on this *"simple"* equation but this equation can be hard to solve. It can be seen that the two body problem is equivalent to a central force problem where the potential is the gravitational force $f(\vec{r}) = -GM/|r|^2$, therefore in order to find an analytical solution it can be written

$$\ddot{\vec{r}} = f(\vec{r})\,\frac{\vec{r}}{|r|}\,. \tag{2.8}$$

At first it is necessary to show that the movement in such a central force is plain in order to simplify the differential equation. To show this, the position vector $\vec{r}$ is multiplied

$$\vec{r} \times \ddot{\vec{r}} = f(\vec{r})\,\frac{\vec{r} \times \vec{r}}{|r|} = 0\,. \tag{2.9}$$

The left hand side of this equation can be written as the derivation of the vector product between position and velocity vector

$$\frac{\partial}{\partial t}\left(\vec{r} \times \dot{\vec{r}}\right) = \left(\vec{r} \times \ddot{\vec{r}}\right) = 0\,. \tag{2.10}$$

So it is shown that the vector product is time independent and can be integrated for t and this yields to a conservation equation

$$\vec{r} \times \dot{\vec{r}} = \vec{C}\,. \tag{2.11}$$

This conservation equation verifies that the motion of the satellite is in a constant plane and that this plane is determined by the position vector $\vec{r}$ and the velocity vector $\dot{\vec{r}}$, it is perpendicular to the vector $\vec{C}$ the integration constant, this is shown in fig. 2.6. The constant vector $\vec{C}$ determines the first necessary elements in order to describe the orbit. It can be seen that the plane of motion is inclined to the

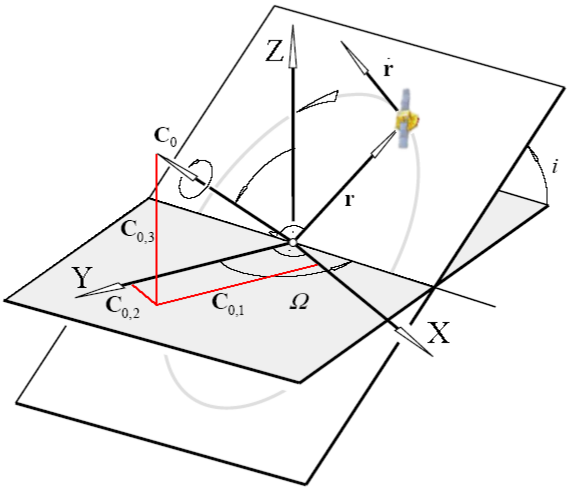


Figure 2.6: Plane Motion; motion in a central force problem

plane which is determined by the unity vector in the X and Y direction. This parameter is the inclination i of the motion. The angle between the unity vector in X direction and the cross section is the right ascension of ascending node Ω (RAAN). The adjustment vector of the motion plane could be written as

$$\vec{C}_0 = \begin{pmatrix} \sin(i)\sin(\Omega) \\ -\sin(i)\cos(\Omega) \\ \cos(i) \end{pmatrix} \begin{pmatrix} e_X \\ e_Y \\ e_Z \end{pmatrix} \tag{2.12}$$

where e_X, e_Y and e_Z are the unity vectors for the axes. This result can be used to simplify equation (2.7) to that effect that just a two dimensional motion has to be considered. This can be reached by using the following transfer matrix and splitting up in a parallel and perpendicular motion (cf. [11] section 2)

$$\begin{pmatrix} \ddot{r}_\parallel \\ \ddot{r}_\perp \\ C_0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(i) & \sin(i) \\ 0 & -\sin(i) & \cos(i) \end{pmatrix} \begin{pmatrix} \cos(\Omega) & \sin(\Omega) & 0 \\ -\sin(\Omega) & \cos(\Omega) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} . \tag{2.13}$$

Using this transformation it is possible to proceed in two dimensions only and write the position vector $\vec{r}$ in rectangular components in the plane of motion (polar coordinates)

$$\vec{r} = |r| \left(\cos(\nu)x + \sin(\nu)y\right) . \tag{2.14}$$

The general equation of motion can be rewritten and yields to

$$\ddot{\vec{r}} \times \vec{C} = -\frac{GM}{|r|^2}\frac{\vec{r}}{|r|} \times \vec{C} = -GM\left(\frac{\vec{r}\dot{\vec{r}}}{|r|^3}\vec{r} - \frac{\vec{r}^2}{|r|^3}\dot{\vec{r}}\right) . \tag{2.15}$$

For the scalar product $\vec{r}\dot{\vec{r}}$ it can be written $\vec{r}\dot{\vec{r}} = |\vec{r}||\dot{\vec{r}}|\cos\left(\vec{r}, \dot{\vec{r}}\right)$. The value $|\dot{\vec{r}}|\cos\left(\vec{r}, \dot{\vec{r}}\right)$ is the projection of the velocity vector on the direction vector. So it can be written

$$\frac{\partial}{\partial t}\left(\dot{\vec{r}}\times\vec{C}\right) = GM\frac{\partial}{\partial t}\left(\frac{\vec{r}}{|r|}\right). \tag{2.16}$$

The integration yields to

$$\dot{\vec{r}}\times\vec{C} = GM\frac{\vec{r}}{|r|}+\vec{A}. \tag{2.17}$$

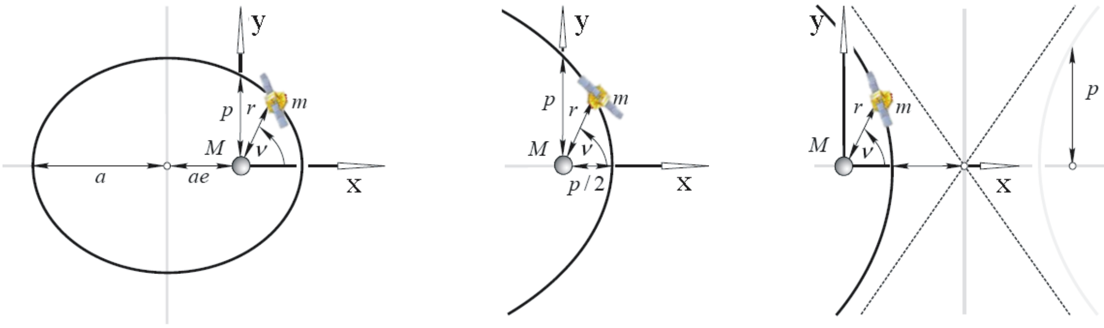
The integration constant $\vec{A}$ is a linear combination of $\vec{r}$ and the vector product $\dot{\vec{r}}\times\vec{C}$ such that $\vec{A}$ lies in the plane of motion and can therefore be written as $\vec{A} = |A|x$. A scalar multiplication with $\vec{r}$ yields to

$$\vec{C}\left(\vec{r}\times\dot{\vec{r}}\right) = GMr\left(1+\frac{A}{GM}\,x\,\frac{\vec{r}}{|r|}\right) \tag{2.18}$$

Rearranging this formula by taken into account that $x\,\vec{r}/|r| = \cos(\nu)$ brings the solution of the general equation of motion and the radial distance between the satellite and the center of mass (CoM) depending on ν

$$\vec{r} = \frac{\vec{C}^2/GM}{1+\frac{A}{GM}\cos(\nu)} = \frac{p}{1+e\cos(\nu)}\,, \tag{2.19}$$

where $p = \vec{C}^2/GM$ and $e = A/GM$. This equation is the polar equation for a conic section with parameter p and eccentricity e. It can be an ellipse, parabola or hyperbola depending on whether e is less than 1, equal to 1 or greater than 1 respectively (see fig. 2.7). Building an orbit propagator on the basis of equation (2.7) and (2.19) is possible (numerical description of such algorithms can be seen in [11, 40]), but impractical as it does not incorporate known regularities of an orbit. Before Newton described the laws of motion in a mathematical way Kepler discovered three basic laws of orbital motions empirically based on conclusions drawn from observations by Tycho Brahe[2]. The next section describes these laws and how they can be proven by Newton's laws.



(a) ellipse $0 \leq e < 1$ (b) parabola $e = 1$ (c) hyperbola $e > 1$

Figure 2.7: Conic Section as a family of solutions for the two body problem

---

[2] danish astronomer/astrologer (*1546-†1601) built Uraniborg one of the first astronomical / astrological observatory in denmark, *"Astronomiae Instauratae Mechanica"*, Wandsbek 1598.

2.3 Kepler's laws

The fundamental properties of orbits are summarised in Kepler's three laws of planetary motion.

1. The orbit of each planet is an ellipse, with the Sun at its focus.
2. The line joining the planet to the Sun sweeps out equal areas in equal times.
3. The square of the period of a planet is proportional to the cube of its mean distance from the Sun.

Although they were stated for planetary motion around the Sun they are equally applicable to satellites around the Earth with one modification. The orbit is not limited to an ellipse, but can be any conical section (cf. section 2.2.2). The laws of Kepler are valid under the assumption that the satellite and the Earth are point masses, and that gravitational forces are the only forces acting on the two bodies. It is also assumed that the two masses are not under influence of gravitational forces from other celestial bodies than each other (this assumption is for a first description applicable because the forces on a satellite from other celestial bodies are minute see [36]). The satellites orbit will, as it is not leaving the Earth gravitational force, be an ellipse with the center of the Earth at one of the foci. In order to get a deeper insight in Kepler's laws the next sections shows how these laws can be proven with Newton's mechanics and how these laws can be used to simplify the general equation of motion.

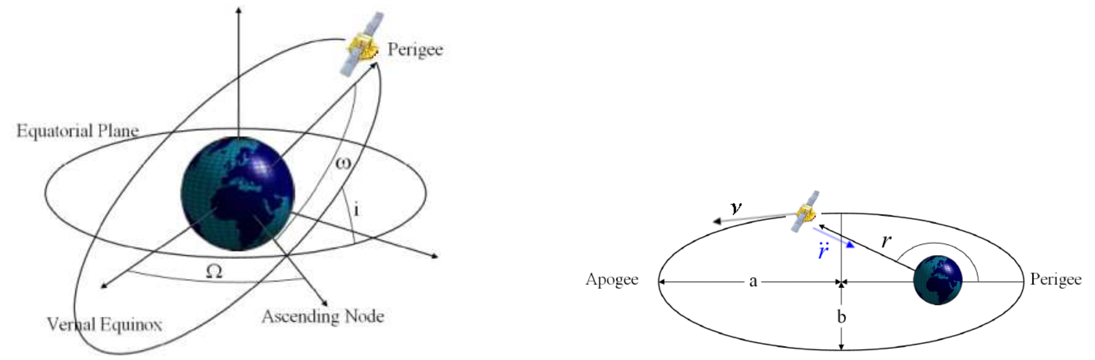
2.3.1 Determination of the orbit shape

All fundamental properties which were stated by Kepler are basic solution of Newton's law of motion (2.7). Kepler's first law is proven by the solution of the two body problem. It states that the radial distance r between the satellite and the CoM depending on ν is an ellipse, parabola or hyperbola. Kepler's first law summarised this in an empirically way.

2.3.2 Conservation of Energy

In order to prove Kepler's second law it is necessary to consider the same assumption made in section 2.2.2, that the satellite with the mass m is much less massive than the Earth's mass M so that the massive body sits in the foci of the elliptical orbit path while the much lighter satellite is in an orbit around it, confer fig. 2.8 (a)/(b). It can be seen that in the circular motion the velocity is tangential to the path taken by the satellite and the acceleration of the satellite is given by equation (2.7), this acceleration $\ddot{\vec{r}}$ is anti parallel to the radial vector. These observations could be used to show that the angular momentum of the motion is conserved. The angular momentum of the system can be written as

$$\vec{L} = \vec{r} \times \left(m \dot{\vec{r}} \right) = m \left(\vec{r} \times \dot{\vec{r}} \right) . \tag{2.20}$$



(a) Orientation of the orbit defined by the right ascension of the ascending node (RAAN) Ω, the Inclination i

(b) 2 dimensional view of the orbit ellipse which is defined by the first Kepler law and proven by the solution of the two body problem, conic section with $0 \leq e < 1$

Figure 2.8: Keplerian Orbit explanations

To show that the angular momentum is conserved it is necessary to look at the time evaluation of $\vec{L}$. According to the assumptions made in section 2.2 there is no additional external torque. So it can be written for the time evaluation

$$\begin{aligned} \frac{\partial}{\partial t}\vec{L} &= \frac{\partial}{\partial t} m \left(\vec{r} \times \dot{\vec{r}}\right) \\ \frac{\partial}{\partial t}\vec{L} &= m\left(\dot{\vec{r}} \times \dot{\vec{r}} + \vec{r} \times \ddot{\vec{r}}\right) \\ &= 0 = \text{ const.} \end{aligned} \tag{2.21}$$

So the angular momentum is constant since the gravitational force is a central force, namely the acceleration points towards the center of the path defined for the satellite by the force and the satellite's velocity. So $\vec{L}$ is conserved and $\vec{L}$ is equal to the integration constant $\vec{C}$ in the considerations above. The vector points to a fixed direction in space, $\vec{r}$ is perpendicular to it. To show that the areas swept out by $\vec{r}$ in equal times are equal, the area dA swept out by the radial vector due to the displacement $\dot{\vec{r}}$ has to be considered

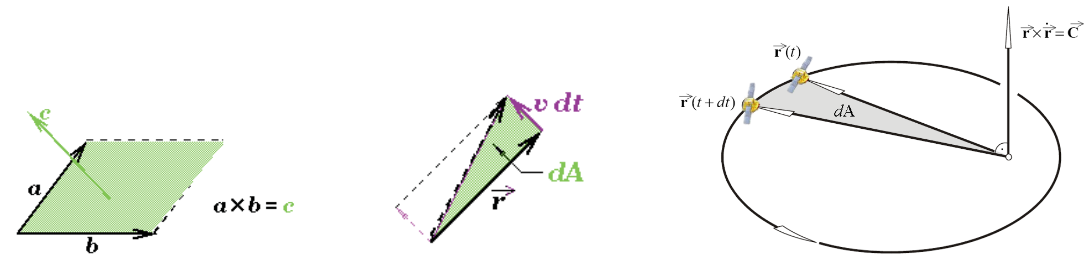
$$dA = \frac{1}{2}\vec{r}(t) \times \vec{r}(t+dt) = \frac{1}{2}\vec{r}(t) \times \left(\vec{r}(t) + \dot{\vec{r}}(t)dt\right) = \frac{1}{2}\vec{r}(t) \times \dot{\vec{r}}(t)dt\,, \tag{2.22}$$

a geometric interpretation is shown in fig. 2.9. The area which is swept out by the radial vector in a given time t_1 and t_2 can be described by the time integral of dA

$$A = \int_{t_1}^{t_2} dA = \frac{1}{2}\int_{t_1}^{t_2} \vec{r} \times \dot{\vec{r}}\,dt \overset{\text{eq. (2.11)}}{=} \frac{1}{2}\vec{C}\left(t_2 - t_1\right)\,. \tag{2.23}$$

So it can be seen that the motional integral (2.11) contains the conserved sweeping velocity.

$$\frac{dA}{dt} = \frac{1}{2}\vec{C} \qquad \text{Kepler's second law}\,. \tag{2.24}$$



(a) general vector product (b) area in a vector product (c) area in an orbit

Figure 2.9: Geometric Interpretation of the Motional Integral for the Angular Momentum

Functionally what this means is that the velocity of the satellite in orbit must be fast when it is near the center of its orbit and slow when it is in the part of its orbit that is further away from the center so that the rate of area swept out is always constant this is parameterised in section 2.3.4. Apart from the angular momentum there is a second conservation equation which can be described by the semimajor axis a of the ellipse, therefore it is necessary to consider the energy in a central gravitational force field in consideration of an energy constant h

$$\frac{1}{2}m\dot{\vec{r}} - m\frac{GM}{|r|} = mh\,. \tag{2.25}$$

The square of the velocity can be written as

$$\dot{\vec{r}}^2 = GM\left(\frac{2}{|r|} + \frac{2h}{GM}\right) = GM\left(\frac{2}{|r|} - \frac{1}{a}\right)\,, \tag{2.26}$$

where $h = -GM/2a$. In this case the semimajor axis of the ellipse has the meaning of an integration constant and describes the total energy of the body in an orbit.

2.3.3 Time derivation of the orbit

Kepler's third law stated that the square of the period of a satellite is proportional to the cube of its mean distance from the Earth. This can be shown by using equation (2.23). The period T of the elliptical orbit can be found by noting that the total area of the ellipse is $A = \pi ab$ where a is the semimajor axis and b is the semiminor axis. Since the areal velocity is constant

$$\frac{dA}{dt} = \frac{A}{T} = \frac{\pi a^2\sqrt{1-e^2}}{T} \tag{2.27}$$

and

$$\frac{dA}{dt} = \frac{1}{2}\vec{C} = \frac{1}{2}\sqrt{GMa\left(1-e^2\right)}\,. \tag{2.28}$$

Therefore combining equations (2.27) and (2.28) it can be written

$$T^2 = \frac{4\pi^2}{GM}a^3 \qquad \text{Kepler's third law.} \tag{2.29}$$

In order to get a deeper insight in the evolution of time considering the motion in the orbit equation (2.11) should be considered with a time derivation of $\vec{r} = |r|\ (\cos(\nu)x + \sin(\nu)y)$, then it can be written

$$\vec{r} \times \dot{\vec{r}} = |r|^2 \dot{\nu}\, x \times y = |r|^2 \dot{\nu} \vec{C}\,. \tag{2.30}$$

Comparing equation (2.11) and (2.30) under consideration that $\vec{r}$ can still be written in polar coordinates this yields to

$$\begin{aligned} \frac{\dot{\nu} p^2}{\left(1 + e\ \cos(\nu)\right)^2} &= C && (2.31) \\ \frac{p^2}{\left(1 + e\ \cos(\nu)\right)^2}\, d\nu &= C\, dt\,. && (2.32) \end{aligned}$$

The indefinitely integration yields to

$$\int\limits_{\nu_0}^{\nu} \frac{p^2}{\left(1 + e\ \cos(\nu)\right)^2}\, d\nu = C\left(t - t_0\right)\ . \tag{2.33}$$

The integral can be solved in a closed from, but it is not possible to get values for every numerical eccentricity e. To solve the integral for orbits with an eccentricity $e < 1$ the eccentric anomaly E has to be introduced [24]. In order to perform this operation it is necessary to use a variable change from $\nu \rightarrow E$ [34, 37], in fig. 2.10 the equation for E can be determined

$$|r| \cos(\nu) = a \cos(E) - ae\,. \tag{2.34}$$

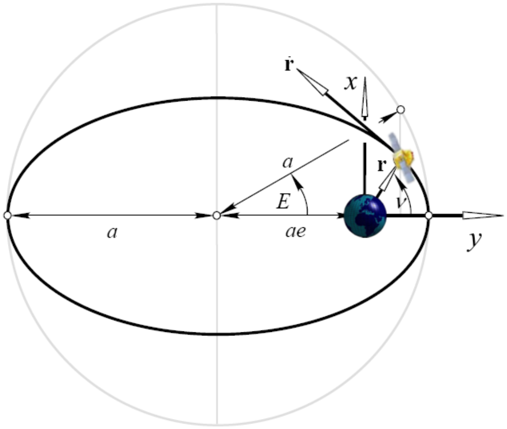


Figure 2.10: Illustration of the Eccentric Anomaly E

By changing $|r|$ with the polar equation of $|r|$ it can be written

$$\frac{p \cos(\nu)}{1 + e \cos(\nu)} = a\left(\cos(E) - e\right)\ . \tag{2.35}$$

Using the relation between a and e for an ellipse it yields to

$$\cos(\nu) = \frac{\cos(E) - e}{1 - e\cos(E)} \tag{2.36}$$

$$\sin(\nu) = \frac{\sqrt{1-e^2}\,\sin(E)}{1 - e\cos(E)}\,. \tag{2.37}$$

It follows that

$$\tan^2\left(\frac{\nu}{2}\right) = \frac{1-\cos(\nu)}{1+\cos(\nu)} = \frac{1+e}{1-e}\,\frac{1-\cos(E)}{1+\cos(E)} = \frac{1+e}{1-e}\tan^2\left(\frac{E}{2}\right) \tag{2.38}$$

or

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}}\,\tan\left(\frac{E}{2}\right). \tag{2.39}$$

This equation determines the true anomaly ν for a given eccentric anomaly. It is sometimes called the *Gauss* equation. By differentiate equation (2.36) and using equation (2.37) it can be written

$$\frac{d\nu}{dE} = \frac{\sqrt{1-e^2}}{1 - e\cos(E)}\,. \tag{2.40}$$

By integrating this equation and combining it with equation (2.33) it is possible to get a connection between the time t and the eccentric anomaly E

$$E - e\sin(E) = \frac{C}{a^2\sqrt{1-e^2}}\,(t - t_0) = \sqrt{\frac{GM}{a^3}}\,(t - t_0) = M\,, \tag{2.41}$$

where M is the mean anomaly. This is Kepler's equation, one of the most celebrated in orbital mechanics. This equation gives the relation between the polar coordinates of a celestial body (here the satellite) and the time elapsed from a given initial point. By solving this transcendental equation[3] it is possible to get for every given time t the eccentric anomaly. The mean anomaly M contains the mean motion n of the satellite

$$M = \sqrt{\frac{GM}{a^3}}\,(t - t_0) = n\,(t - t_0)\,. \tag{2.42}$$

This means when a satellite in a central force field would orbit around the CoM with such mean velocity the satellite would achieve the same exact position after one orbit as before. In contrast to this mean velocity it is necessary to calculate the real angular velocity in order to get a precise orbit propagation, therefore it can be written

$$\dot{\nu} = \frac{C}{p^2}\,(1 + e\cos(\nu))^2 \tag{2.43}$$

the true angular velocity is thus changing according to the true anomaly ν only for $e = 0$ the velocity is constant.

The general consideration of the Kepler problem shows that there are six different

---

[3] Transcendental equations cannot be solved in a closed form, thus solving means using a numerical iterative method. An exact solution was developed by Lagrange in the form of the trigonometric series (see [37]).

integration constants which are necessary to propagate the orbit. These integration constant should be defined in detail in the next section.

2.3.4 Keplerian Elements

In the previous sections it was shown that an orbit of an artificial satellite can be determined by using basic equations from celestial mechanics. The closed form solution of these formulae are often difficult to achieve or even impossible. That is why in order to create an orbit propagation tool it is necessary to simplify these formulae. This is possible by using geometrical characteristic of the orbit. These characteristics appeared in the sections above as the integration constants these constants specify completely the properties of an elliptical orbit.

Symbol	Name	Meaning
Semimajor axis	a	Shape of the Orbit
Eccentricity	e	Shape of the Orbit
Inclination	i	Orientation of the Orbit
Right ascension of ascending node	Ω	Orientation of the Orbit
Argument of Perigee	ω	Orientation of the Orbit
Epoch	t_0	Time of Perigee
Mean Anomaly	$M = n\,(t - t_0)$	Position in the Orbit

Table 2.1: Keplerian Elements

If only the gravitational force as assumed by Newton's law is acting on the satellite the first five parameters are constant and the orbit is an ideal Keplerian orbit (see orbit calculations in [6]). Only the true anomaly of the satellite changes its value due to the undisturbed movement in the orbit. If disturbance forces are present the motion becomes an osculating orbit. Due to the disturbance forces the shape and the orientation of the orbit will change. Thus the Keplerian elements will change their values. If the orbital motion then should be described by the Keplerian elements, these elements now become time-variant (details see appendix A.3).

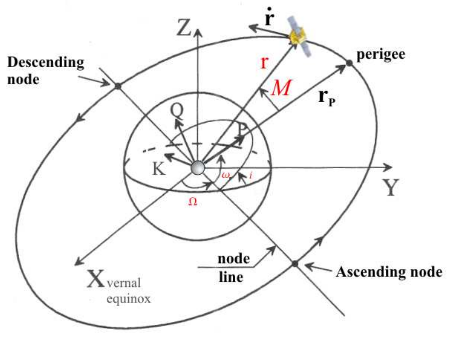


Figure 2.11: Keplerian Elements, parameters defining the position of a satellite in orbit

These six integration constants are defined for a specific point in time t_0, known as the epoch time. The position of the satellite in orbit is then determined using these values as reference (see fig. 2.11). The semimajor axis a is an expression of the size of the orbit, or altitude of the satellite and can be derived from the mean motion n and the eccentricity e. The eccentricity describes the shape of the ellipse (cf. fig. 2.7). The inclination i represents the tilt angle in degrees of the orbital plane with respect to the Earth's equatorial plane (further details see section 2.5). The RAAN Ω is the angle from the vernal equinox defined as the X-axis in the geocentric inertial system to the ascending node, defined as the point where the orbit intersects with the equatorial plane. The argument of perigee ω is defined by the angle between the aforementioned ascending node and the perigee of the orbit. The mean anomaly M describes the position of the satellite in its orbit relative to the perigee. At perigee the mean anomaly is zero increasing to 180° at apogee then back to perigee at 360°. Figure 2.11 shows these elements in an intermediate reference frame known as a perifocal coordinate system. This is an orthogonal axis system with the orbital plane as the fundamental plane and its origin at the Earth's center. The P-axis points towards perigee, the Q-axis is also in the orbit plane in the direction of the satellite motion and the K-axis completes the right-hand set. Using this frame of reference and the mathematics which is stated above it is possible to calculate the the precise position $\vec{r}$ and the velocity $\dot{\vec{r}}$ of the satellite. This is demonstrated in the following flow-chart.

Calculation of the mean angular velocity n

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

Determination of E at time t on the base of Kepler's equation (2.41)

$$n\left(n - t_0\right) = E - e\sin\left(E\right)$$

Determination of the true anomaly ν for time t

$$\cos\left(\nu\right) = \frac{\cos\left(E\right) - e}{1 - e\cos\left(E\right)} \qquad \sin\left(\nu\right) = \frac{\sqrt{1-e^2}\,\sin\left(E\right)}{1 - e\cos\left(E\right)}$$

or with

$$\tan\left(\frac{\nu}{2}\right) = \sqrt{\frac{1+e}{1-e}}\,\tan\left(\frac{E}{2}\right)$$

Calculation of r and the integration constant C

$$r = \frac{a\left(1-e^2\right)}{1+\cos\left(\nu\right)} \qquad C = \sqrt{GM\,p}$$

Determination of the perifocal unity vectors

$$\vec{P} = \begin{pmatrix} \cos\left(\Omega\right)\cos\left(\omega\right) - \sin\left(\Omega\right)\sin\left(\omega\right)\cos\left(i\right) \\ \sin\left(\Omega\right)\cos\left(\omega\right) + \cos\left(\Omega\right)\sin\left(\omega\right)\cos\left(i\right) \\ \sin\left(\omega\right)\sin\left(i\right) \end{pmatrix}$$

and

$$\vec{Q} = \begin{pmatrix} -\cos\left(\Omega\right)\sin\left(\omega\right) - \sin\left(\Omega\right)\cos\left(\omega\right)\cos\left(i\right) \\ -\sin\left(\Omega\right)\sin\left(\omega\right) + \cos\left(\Omega\right)\cos\left(\omega\right)\cos\left(i\right) \\ \cos\left(\omega\right)\sin\left(i\right) \end{pmatrix}$$

Determination of position $\vec{r}$ and velocity $\dot{\vec{r}}$

$$\begin{pmatrix} \vec{r} \\ \dot{\vec{r}} \end{pmatrix} = \begin{pmatrix} r\,\cos\left(\nu\right) & r\,\sin\left(\nu\right) \\ -\frac{C}{p}\,\sin\left(\nu\right) & \frac{C\left(\cos\left(\nu\right)+e\right)}{p} \end{pmatrix} \begin{pmatrix} \vec{P} \\ \vec{Q} \end{pmatrix}$$

or

$$\vec{r} = \vec{R}\left(\Omega,\,i,\,\omega\right)\,\vec{q}\left(a,\,e,\,M\right) \qquad \dot{\vec{r}} = \vec{R}\left(\Omega,\,i,\,\omega\right)\,\dot{\vec{q}}\left(a,\,e,\,M\right)$$

where

$$\vec{R} = \begin{pmatrix} P_1 & Q_1 & C_{0,1} \\ P_2 & Q_2 & C_{0,2} \\ P_3 & Q_3 & C_{0,3} \end{pmatrix} \qquad \vec{q} = \begin{pmatrix} a\,\left(\cos\left(E\right) - e\right) \\ a\,\sqrt{1-e^2}\,\sin\left(E\right) \\ 0 \end{pmatrix} = \begin{pmatrix} r\,\cos\left(\nu\right) \\ r\,\sin\left(\nu\right) \\ 0 \end{pmatrix}$$

and

$$\dot{\vec{q}} = \frac{na}{1-e\cos\left(E\right)} \begin{pmatrix} -\sin\left(E\right) \\ \sqrt{1-e^2}\,\cos\left(E\right) \\ 0 \end{pmatrix} = \frac{na}{\sqrt{1-e^2}} \begin{pmatrix} -\sin\left(\nu\right) \\ e + \cos\left(\nu\right) \\ 0 \end{pmatrix}$$

or vice versa (calculation of the Keplerian elements)

Determination of the inclination i an the RAAN Ω

$$\vec{r} \times \dot{\vec{r}} = \vec{C} \qquad \rightarrow \qquad C_0 = \begin{pmatrix} \sin(i)\sin(\Omega) \\ -\sin(i)\cos(\Omega) \\ \cos(i) \end{pmatrix} \rightarrow \qquad i,\, \Omega$$

$\downarrow$

Determination of a semimajor axis

$$a = \frac{1}{\frac{2}{|r|} - \frac{\dot{\vec{r}}^2}{GM}}$$

$$\text{ellipse: } a > 0 \rightarrow \frac{2}{|r|} > \frac{\dot{\vec{r}}^2}{GM} \text{ circle: } \dot{\vec{r}}^2 = \frac{GM}{|r|}$$

$$\text{parabola: } a = \infty \rightarrow \frac{2}{|r|} = \frac{\dot{\vec{r}}^2}{GM} \text{ hyperbola: } a < 0 \rightarrow \frac{2}{|r|} < \frac{\dot{\vec{r}}^2}{GM}$$

$\downarrow$

Determination of the numeric eccentricity e

$$p = \frac{C^2}{GM} \qquad e = \sqrt{\frac{a-p}{a}}$$

$\downarrow$

Determination of the true anomaly ν for time t

$$\cos(\nu) = \frac{p-r}{er} \qquad \sin(\nu) = \frac{p(\vec{r}\dot{\vec{r}})}{erC}$$

$\downarrow$

Determination of the perifocal unity vectors

$$\begin{pmatrix} \vec{P} \\ \vec{Q} \end{pmatrix} = \begin{pmatrix} \frac{e+\cos(\nu)}{p} & -\frac{r\sin(\nu)}{C} \\ \frac{\sin(\nu)}{p} & \frac{r\cos(\nu)}{C} \end{pmatrix} \begin{pmatrix} \vec{r} \\ \dot{\vec{r}} \end{pmatrix}$$

$$K = \cos(\Omega)X + \sin(\Omega)Y$$

$\downarrow$

Determination of ω

$$\cos(\omega) = \vec{K}\vec{P} \qquad \sin(\omega) = -\vec{K}\vec{Q}$$

$\downarrow$

Determination of the eccentric anomaly E for time t

$$\cos(E) = \frac{\vec{r}\vec{P}}{a} + e \qquad \sin(E) = \frac{\vec{r}\vec{Q}}{a\sqrt{1-e^2}}$$

$\downarrow$

Determination of Epoch time t_0

$$M = e - e\sin(E) \qquad t_0 = t - M\sqrt{\frac{a^3}{GM}}$$

However, the Keplerian elements and with them the Keplerian orbit model is an ideal one, based on the assumption that the motion of the satellite is a result of the gravitational attractions between two bodies. For a first approximation it is reasonable to make this assumptions, but for precise orbit calculations other forces must be taken into account. Although these forces are sometimes minute (appendix A.5 and [36, 43]) in comparison to the main force which determines the orbit they are still present and could lead to errors in the orbital calculations.

In order to avoid such errors in the propagation of satellites the North American Aerospace Defense Command (NORAD) maintain general information about the perturbation on all space objects and a Simplified General Perturbation Model (SGP) developed by C.G. Hilton and J.R. Kuhlman (1966) described in the Spacetrack Report No. 3 [20] (see also [9]). This perturbation elements and the SGP model are described in the next section and used by **VIS SAT** . Although these complex models are based on the first principles in celestial mechanics.

2.4 NORAD Orbit Propagation

NORAD maintains general perturbation element sets on all resident space objects. These element sets are periodically refined so as to maintain a reasonable prediction capability on all space objects. In turn, these element sets are provided to users, providing them with a means of propagating these element sets in time to obtain a position and velocity vector of the space object (**VIS SAT** includes the data sets published in 2005 in a `txt`-file format).

The most important point to be noted is that not just any prediction model will suffice. The NORAD element sets are mean values obtained by removing periodic variations in a particular way. In order to obtain accurate predictions, these periodic variations must be reconstructed (by the prediction model) in exactly the same way they were removed by NORAD. Hence, inputting NORAD element sets into a different model (even though the model may be more accurate or even a numerical integrator) will result in degraded predictions. The NORAD element sets must be used with these models in order to retain maximum prediction accuracy. All space objects are classified by NORAD as near-Earth (period less than 225 minutes) or deep-space (period greater than or equal 225 minutes). Depending on the period, the NORAD element sets are automatically generated with the near-Earth or deep-space model.

2.4.1 NORAD Two-Line Orbital Element Set (TLE)

NORAD uses a generalised form of data sets, the data for each satellite consists of three lines in the following format

```
AAAAAAAAAAAAAAAAAAAAAAAA
1 NNNNNU NNNNNAAA NNNNN.NNNNNNNN +.NNNNNNNN +NNNNN-N +NNNNN-N N NNNNN
2 NNNNN NNN.NNNN NNN.NNNN NNNNNNN NNN.NNNN NNN.NNNN NN.NNNNNNNNNNNNNN
```

Line 0 is a twenty-four character name (to be consistent with the name length in the NORAD SATCAT). Lines 1 and 2 are the standard two-line orbital element set format. The format description is the following (cf. [12]).

Line 1

Column	Description
`01`	Line Number of Element Data
`03-07`	Satellite Number
`08`	Classification (U=Unclassified)
`10-11`	International Designator (Last two digits of launch year)
`12-14`	International Designator (Launch number of the year)
`15-17`	International Designator (Piece of the launch)
`19-20`	Epoch Year (Last two digits of year)
`21-32`	Epoch (Day of the year and fractional portion of the day)
`34-43`	First Time Derivative of the Mean Motion
`45-52`	Second Time Derivative of Mean Motion (decimal point assumed)
`54-61`	BSTAR drag term (decimal point assumed)
`63`	Ephemeris type
`65-68`	Element number
`69`	Checksum (Modulo 10)

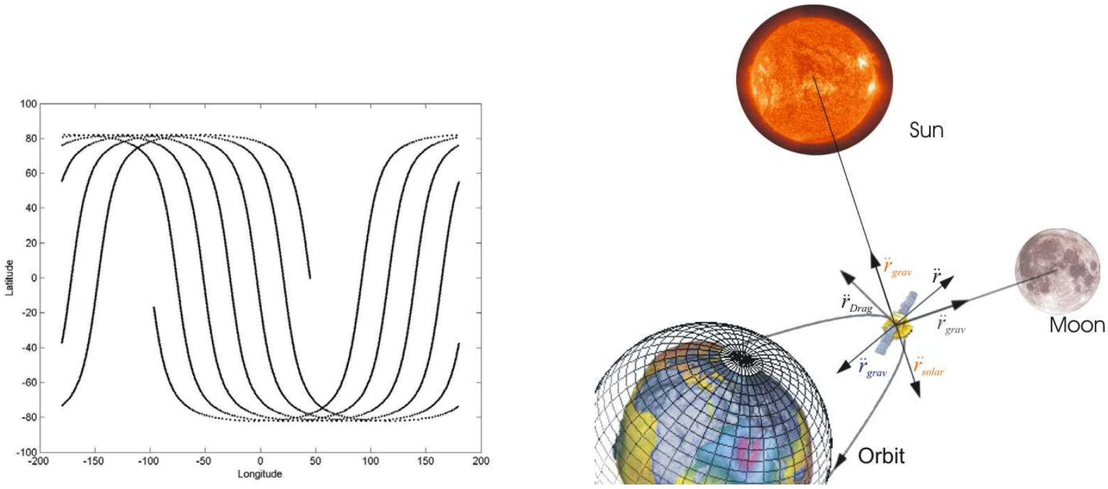
Line 2

Column	Description
`01`	Line Number of Element Data
`03-07`	Satellite Number
`09-16`	Inclination [Degrees]
`18-25`	Right Ascension of the Ascending Node [Degrees]
`27-33`	Eccentricity (decimal point assumed)
`35-42`	Argument of Perigee [Degrees]
`44-51`	Mean Anomaly [Degrees]
`53-63`	Mean Motion [Revs per day]
`64-68`	Revolution number at epoch [Revs]
`69`	Checksum (Modulo 10)

This generalised data is produced for the SGP models and only with this models it is possible to obtain accurate propagation of the orbit. It can be seen that the first line contains various identification data sets and also the perturbation influences by the geopotential and the drag forces. The second line contains the Keplerian elements (cf. section 2.3.4) to describe the orbital plane, the only parameter which is missing is the semimajor axis a but this parameter can be determined by the mean motion (see equation (2.42)). The next section describes the SGP model and how the Kepler calculations are contained in this model.

2.4.2 Simplified General Perturbation Model

Given the Keplerian elements for a single point in time, the estimation of the future position becomes straight forward (refer to flow chart on pages 17 and 18). However, it should be mentioned that the assumptions on which Keplers laws are based are not accurate, certain improvements utilising known irregularities can be made. The biggest source of error is due to the fact that the Earth is not being perfectly circular. The deformation is often parameterised by the geopotentional function as described in appendix A.1 (see also [56]). The deformation can be described by using the zonal harmonic coefficients J_i for i'th order deformations. Gravitational forces from the Sun and the Moon, tidal Earth and ocean, and different electromagnetic radiations, also have more or less influence on the perturbations of a satellite orbit confer appendix A.5. The perturbations are usually divided into secular, short period, and long period perturbations. The short period perturbations are periodic with a period shorter than the satellites orbital period, while the long period perturbations have a longer period than the satellite. Secular perturbations are those that cause elements to steadily diverge over time. Periodic perturbations are those that impart a sinusoidal variation in elements over time. Figure 2.12 (a) shows a simple orbit propagation of Keplerian elements plotted with latitude against longitude in comparison figure 2.12 (b) shows how are the elements change due to the different perturbations.



(a) undisturbed Keplerian Orbit (b) perturbated Kepler elements

Figure 2.12: Keplerian Orbit with Perturbations

This section describes the additional perturbations and how they are considered in the models, only the essential formulae are shown here for details see appendix A.4.

2.4.2.1 Perturbations due to Geopotential

The Earth is not spherical, in fact it has a bulge at the equator, is flattened at the poles and is slightly pear-shaped. This leads to perturbations in all Keplerian elements. The second order deformation of the Earth considers the fact that it is slightly flattened. According to the Lagrange planetary equations, the flattening factor, J_2, results in the following time derivatives of the right ascension of ascending node, and the argument of perigee (cf. appendix A.4.1).

$$\frac{\partial \Omega}{\partial t} = -\frac{3}{2} J_2 \frac{R_E^2}{p_0^2} n_0 \cos(i_0) \tag{2.44}$$

$$\frac{\partial \omega}{\partial t} = \frac{3}{4} J_2 \frac{R_E^2}{p_0^2} n_0 \left(5\cos^2(i_0) - 1\right) \tag{2.45}$$

where R_E is the Earth radius, and the numerical value of J_2 for the Earth is $1.08284 \cdot 10^{-3}$. Both of these first order time derivatives are smallest for circular orbits, $e = 0$, but have their minimum for different inclinations.

$$i_{\min,\dot{\Omega}} = \cos^{-1}(0) = 90^\circ \tag{2.46}$$

$$i_{\min,\dot{\omega}} = \cos^{-1}\sqrt{\frac{1}{5}} \approx 63.43^\circ \text{ or } 116.57^\circ \tag{2.47}$$

2.4.2.2 Perturbations due to the Sun and the Moon

The Sun and the Moon causes periodic variations in all Keplerian elements, but secular perturbations only to the right ascension of ascending node and the argument of perigee. For nearly circular orbits, an approximation suggested in [56] yields to

$$\text{Sun: } \frac{\partial \Omega}{\partial t} = -1.54 \cdot 10^{-3} \frac{\cos(i_0)}{k} \tag{2.48}$$

$$\text{Moon: } \frac{\partial \Omega}{\partial t} = -3.38 \cdot 10^{-3} \frac{\cos(i_0)}{k} \tag{2.49}$$

and

$$\text{Sun: } \frac{\partial \omega}{\partial t} = 0.77 \cdot 10^{-3} \frac{5\cos^2(i_0) - 1}{k} \tag{2.50}$$

$$\text{Moon: } \frac{\partial \omega}{\partial t} = 1.69 \cdot 10^{-3} \frac{5\cos^2(i_0) - 1}{k} \tag{2.51}$$

where k is the number of revolutions per day and $\dot{\Omega}$ and $\dot{\omega}$ are given in degrees per day (cf. [31]). These perturbations have their minima for the same inclinations, i, as the non spherical Earth perturbations and become larger for higher altitude orbits.

In accordance with these perturbations the Keplerian elements have to be modified but the basic calculations are still valid. The orbit can be defined by using the steps shown in the flow chart on page 17.

2.4.3 Simplified General Perturbation Model Version 4

The SGP4 model was an extension of the SGP model in order to get a more precise estimation new aspects of the perturbations are included (details see appendix A.4.2, for the deep space version see appendix A.4.3). The atmospheric drag could be estimated by using the ballistic coefficient B^* in the TLE data sets and a term for the perturbations due to solar radiation.

2.4.3.1 Perturbations due to Atmospheric Drag

The atmospheric drag is a force creating an acceleration, $\ddot{\vec{r}}_{\text{Drag}}$, in the opposite direction of the satellites velocity.

$$\ddot{\vec{r}}_{\text{Drag}} = -\frac{1}{2} q_0 \left(\frac{c_w A}{m}\right) \dot{\vec{r}}^2 = -\frac{1}{2} q_0 B^* \dot{\vec{r}}^2 \tag{2.52}$$

where q_0 is the density of the atmosphere, c_w is the drag coefficient, A the cross section area and m the mass of the satellite. The atmospheric drag is a breaking force and hence removes energy from the satellite in orbit. This leads to a decrease in orbital height, but at very low rates.

2.4.3.2 Perturbations due to Solar Radiation

The acceleration caused by solar radiation has a magnitude of

$$\ddot{\vec{r}}_{\text{solar}} = -4.5 \cdot 10^{-6} \left(1 + \chi\right) \frac{A}{m} \tag{2.53}$$

where χ is a reflection factor between 1 and 0. The perturbations due to solar radiation is in the same magnitude as atmospheric drag perturbations for altitudes at 800 km, and less for lower orbits.

The purpose of this SGP model in combination with the TLE sets is to propagate the orbit of a satellite by using the mathematics described in section 2.3.4. The SGP4 utilises the way in which the TLE was constructed. This means that the periodic variation, and the way that they were removed, is taken into consideration when the orbit is reconstructed and propagated. The SGP4 model reconstructs all periodic perturbations, both the short period ones and the long period ones. For orbits with a period longer than 225 minutes additional deep-space perturbations have to be considered, this is done in the SDP4 model (see appendix A.4.3). These perturbation models are implemented in **Vis Sat** and the TLE sets are also included (newer TLE data sets can be obtained by subscribing on the TLE data set web page: http://www.space-track.org). Taken this mathematical bases into account, the next section shows typical satellite orbits and what are their characteristics.

2.5 Satellite Orbits

The orbit of a satellite is described by the Keplerian elements and its orbital motion by the consideration of all forces acting on the satellite. The main motion results of a balance between the gravitational force and the centripetal force which is directly proportional to the velocity $\dot{\vec{r}}$ of the satellite. In order to get a deeper insight of the satellite dynamics and which orbits are characterised by the Keplerian elements, this section describes different orbits and what are the distinctive characteristic Keplerian elements.

All orbital motion around the planet could be summarised in three groups of orbit types.

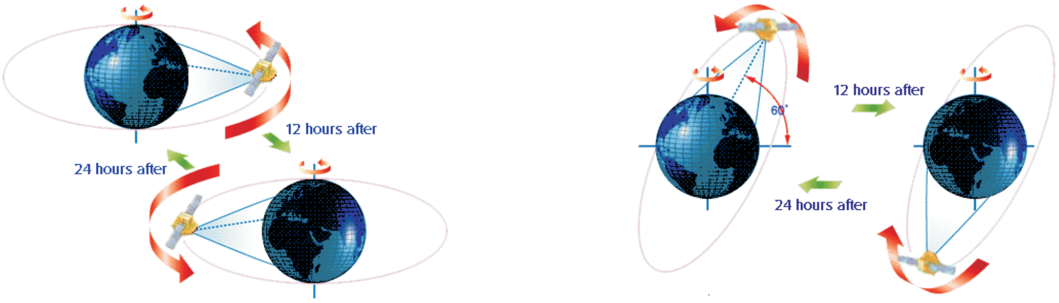
2.5.1 Synchronous Orbits

When the satellites are synchronous, it is necessary to declare the reference frame in which they are synchronous. These orbits can be subdivided into two different orbits, on the one hand the Earth synchronous orbits and on the other hand the Sun synchronous orbits.

2.5.1.1 Earth Synchronous

Earth synchronous orbits known as geostationary orbits, satellite in these orbits circle the Earth at the same rate as the Earth spins. The Earth actually takes 23 hours, 56 minutes and 4.09 seconds to make one full revolution (one sidereal day 23.9344696 hours). Due to Kepler's law, the orbit can be described with an eccentricity $e \approx 0$ and an inclination $i \approx 0°$ which means that it is above the Earth's equator. The altitude is about 35,790.00 km $\approx 6.62\ R_E$ and the satellite travels at 3 km/s. In this case the satellite will appear to hover in the same place in the sky to a stationary observer on the Earth's surface (cf. fig. 2.13 (a)).

Another orbit in which the satellite is synchronous in the reference frame of the Earth is when its completes one circuit around the Earth in one day and appears in the same position above the Earth surface. The duration of one orbit of the satellite is the same as the Earth's rotation period. The inclination and eccentricity of such orbits could differ from zero but the eccentricity will never be bigger than 0.3 (cf. fig. 2.13 (b)).



(a) Geostationary Orbit (b) Synchronous Orbit

Figure 2.13: Earth Synchronous Orbits

2.5.1.2 Sun Synchronous

In a Sun synchronous orbit the direction of the rotation of the orbital plane and the period are the same as the Earth's orbital period. These orbits allows a satellite to pass a section of the Earth at the same time of the day. These satellites have a revolution time of one year and the orbital plane and the orientation of the Sun are always the same. These orbits have an inclination about 80°. The altitude normally lies between 700 and 800 km.

2.5.2 Polar Orbits

In a polar orbit the satellite generally flies at low altitudes and passes over the planet's poles on each revolution. The polar orbit remains fixed in space as the Earth rotates inside the orbit. The inclination of these orbits therefore is $i \approx 90°$.

2.5.3 Recurrent Orbit

A recurrent orbit is an orbit in which the satellite returns to the same position over the surface of the Earth within 24 hours, regardless how many orbit is has made in that time. The orbital period of the satellite is an integral fraction of the Earth's rotation period. If the perigee is about 600 km and the apogee about 40,000 km the satellite will follow an elongated elliptical orbit with a period of about 12 hours (cf. fig. 2.14).

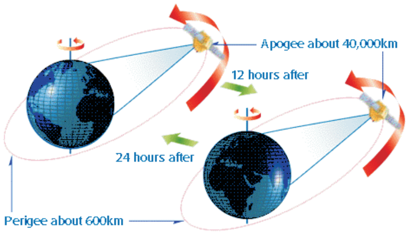


Figure 2.14: Recurrent Orbit

2.6 Summary

The consideration made in this section are to get a deeper insight in the mathematical equations and methods in order to describe a satellite's motion in orbit around the Earth. The focus has been on outlining the origins of the theory and the application of these theories in a model. With the consideration of Newton's laws and Kepler's empirical observations and including perturbations due to the different forces which acts on a satellite, it is possible to create a numerical propagation tool. In order to describe the motion of a satellite the basic equations for a body in a central force field was analysed and the two body problem was solved analytically. The solution of the two body problem are the empirical observations made by Kepler. The description of the orbit by using Kepler's laws yields to a parameterised way to characterise an orbit. The Keplerian elements are six parameters who characterise an orbit precisely. By knowing these parameters it is analytical possible to calculate the exact position of a satellite in an ideal central force field. Due to the fact that the satellite's motion is not in an ideal field, different perturbations have to be considered. The SGP4/SDP4 model published by NORAD in 1988 provides such a numerical model to take the different perturbations due to the effect of gravity, atmospheric drag and solar radiation into account. The satellite's dynamic processes and calculations are implemented in the functions `SatPOS` and `SatTRACK` of the **Vis Sat** toolbox.

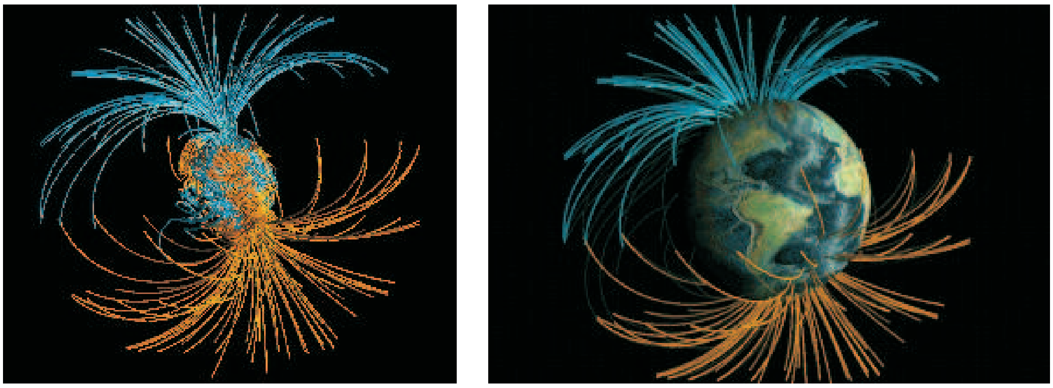
3 Magnetic Coupling Model

Vis Sat provides in addition to the orbit propagation also the functionality to study the magnetic conjunction between the ground and the satellite. Therefore, the satellite trajectory (here the position vector $\vec{r}$) which is calculated by using the mathematic equations described above, is used as input parameter for the Tsyganenko model [49] - [54] of the Earth's magnetic field. The model is developed by Andrei Nikolai Tsyganenko (`Nikolai.Tsyganenko@gsfc.nasa.gov`) and under [55] the FORTRAN code for Tsyganenko's `GEOPACK` can be found. The `GEOPACK` library consists of subsidiary FORTRAN subroutines for magnetospheric modeling studies, including the current International Geomagnetic Reference Field (IGRF) and past Definitive Geomagnetic Reference Field (DGRF) internal field models, a group of routines for transformations between various coordinate systems and a field line tracer. These FORTRAN routines have been translated to MATLAB® by Paul O'Brien (`Paul.OBrien@aero.org`) and have been modified for **Vis Sat** , the present set of subroutines is intended as a subsidiary package for calculating the geomagnetic field components at any point of space within the Earth's magnetosphere up to the Moon's orbit. Upon specifying the universal time and day of year as input parameters, it automatically performs all the necessary rotations and specifications needed. The newest edition of `GEOPACK` which is official released by NASA and which is approved is the 2003 version of the `GEOPACK`, nevertheless under [55] the newest version 2005 is available but still in the test phase. That is why here the version 2003 is used, for future work it is then necessary to replace the `GEOPACK` routines with the newer versions. In order to ensure the compatibility with **Vis Sat** , the toolbox contains a `GEOPACK.m` file as a description of the used interface functions (details [55]). With this description it is possible to modify the future releases of `GEOPACK` due to the **Vis Sat** internal structure which is shown in the schematic on page 35 and described in section 4.

This section gives a briefly introduction into the IGRF and Tsyganenko's model in order to explain the mathematics and the simplifications made. The detailed MATLAB® files for the translation from the original FORTRAN files are included on the attached CD-Rom.

3.1 Earth Magnetic Field

Magnetic field models can be utilised to predict advantageous constellations of satellites and their conjunctions with ground stations but therefore it is necessary to define the Earth's magnetic field. In order to define this field in a mathematical way it is necessary to have a closer look what causes the magnetic field and then how it could be determined. The main cause of the magnetic field is the inner structure of the Earth's core and the related processes. The inner core consists of fluid iron, these iron is due to the effect of convection circulating and together with the spin motion of the Earth the iron forms streams parallel to the Earth's axis (known as α-effect). In addition to these parallel streams there is a turbulent flow of iron which is anti parallel to the Earth axis (known as ω-effect). So in this case, the convection drives the outer core fluid and it circulates relative to the Earth. This means the electrically conducting material moves relative to the Earth's magnetic field. If it can obtain a charge by some interaction like friction between layers, an effective current loop sustains the magnetic dipole type field of the Earth. It is the self induction of these circulations in combination with the magnetic field which keeps the field up. These effects are called the geodynamo (see details [10]) who is the cause of the magnetic field which reaches out into space. In a first approximation, this field could be described as a dipole field but due to the effect of the turbulent streams the dipole field gets disturbed. The processes in the inner core could be described by equations from hydrodynamics and combining them with magnetostatic physical laws (see [10]). These consideration could be summerised in a numerical magneto hydrodynamics (MHD) model which describes the shape of the magnetic field without perturbations due to the effect of the solar wind. This was done at the Pitsburgh Supercomputer Institute by G. Glatzmaier [15] and can be seen in fig. 3.1



(a) 3D magnetic field lines (inside) (b) 3D magnetic field lines (dipole character)

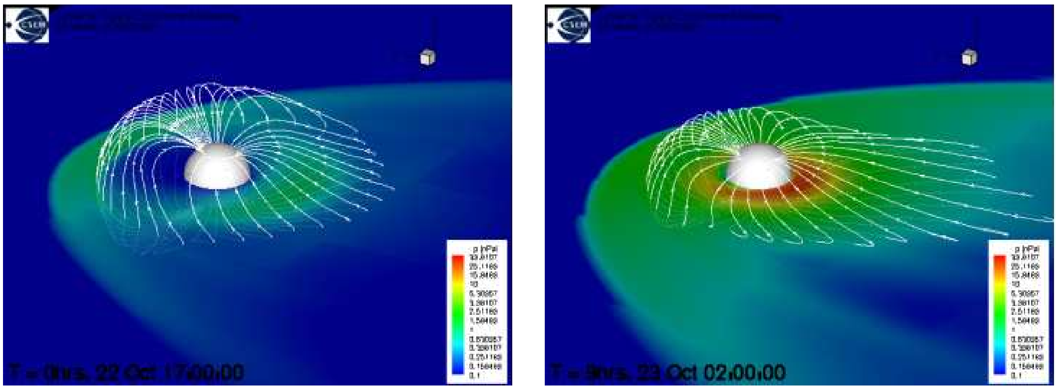
Figure 3.1: Three Dimensional Structure of the Earth's Magnetic Field (source: [15])

It can be seen that the shape of the magnetic field lines which reach out in space have nearly a dipole character. Such a field could be described from the Earth's surface by

$$\vec{B}_{r,\theta} = -\frac{\mu_0 m_{\mathrm{dm}}}{4\pi r^3} \begin{pmatrix} 2\cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{3.1}$$

where $m_{\mathrm{dm}} = 7.94 \cdot 10^{22}$ Am^2 is the magnetic dipole moment, $\mu_0 = 4\pi \cdot 10^{-7}$ m kg / A^{-2}s^{-2} is the magnetic permeability. But due to the effect of the solar wind pressure and the current system which is produced by the magnetic field, the field is far from static and therefore its average variations also need to be described, e.g. effects of varying solar pressure of the solar wind and the angle between the Earth's dipole axis and the Sun-Earth direction. All these effects changes the shape of the dipole field and makes it difficult to describe the field in an exact mathematical framework. A sketch of the dipole field with perturbations is shown in fig. 3.2.

It can be seen that due to the fact of perturbations the dipole characteristics is changed dramatically. On the day side the field is compressed to a few Earth radii and on the night side the field is stretched to a magnetic tail with a range of a few hundred Earth radii. Therefore it is necessary to find a model to describe this structure. The IGRF model is such a model which describes the magnetic field empirically. It is recommended for scientific use by the International Association of Geomagnetism and Aeronomy (IAGA) from the year 2000 and forward. The model consist of a series of truncated spherical harmonic series describing the main field



(a) dipole field with perturbations under low solar wind conditions

(b) dipole field with perturbations under high solar wind conditions

Figure 3.2: Earth's Dipole Field under different outer Conditions (source: [8])

and its variations. It is based on empirical data from several different sources such as land based measurements, ships, planes and satellites [23]. The magnetic field $\vec{B}$ is described in the IGRF model by the gradient of the scalar potential V

$$V = R_E \sum_{n=1}^{N} \sum_{m=0}^{n} \left(\frac{R_E}{r} \right)^{n+1} \left(g_n^m \cos(m\phi) + h_n^m \sin(m\phi) \right) \vec{P}_n^m \cos(\theta) \tag{3.2}$$

where R_E is the mean radius of the Earth, ϕ is the latitude and $\theta = 90° - \phi$ the colatitude and $\vec{P}_n^m \cos(\theta)$ are the quasi Schmidt- normalised associated Legendre functions (see appendix A.6.1). The coefficients g_n^m and h_n^m are spherical harmonic coefficients determined by the empirical measurements. The main contribution in the spherical harmonic expansion arises from the terms with $n = 1$. It can be identified with the field produced by a dipole (cf. fig. 3.1) with center coininciding with the center of the Earth and a dipole axis which is inclined with respect to the polar axis. The contributions of the other terms can be considered to be perturbations of the main dipole field. So in this case this model enables the user to obtain estimates of the strength and direction of the internally generated terrestrial magnetic field at a given location, including information such as L-value (the equatorial crossing distance of a given field line) for given values of latitude, longitude, altitude and time (epoch).

The Tsyganenko model extends the IGRF by including magnetic field contributions from both internal and external mechanisms. These include realistic definitions of the magnetopause and large-scale current systems and handling of IMF variations. The model is essential for the mapping of field lines in the distant magnetosphere, in this thesis it is used for ground satellite conjugate studies. The model is described in the next section.

3.2 Tsyganenko's Magnetic Field Model

Tsyganenko developed a geomagnetic model in order to create a realistic representation of the Earth's internal magnetic field (e.g. DGRF or IGRF) and the total magnetic field in the magnetosphere. The form of the magnetosphere is a combination of the self induced field due to the circulations in the inner core and a current system high above the Earth's surface. This current system is mainly influenced by the Sun's activity. The Sun emits a steady stream of particles as it burns its nuclear fuel. These high energetic atomic particles (electrons and protons) are charged, giving the solar wind an associated magnetic field, which interacts with the Earth's own magnetic field by producing a current system (see details [38]). In the section above it was described, that the magnetic field shape without perturbation can be assumed as a dipole field. According to this assumptions Mead and Fairfield (1975) developed a first numerical model of the geomagnetic field based on large sets of magnetic field data to include the perturbations. Based on this model Tsyganenko and Usmanov developed an advanced empirical magnetic field model of the Earth's magnetosphere using in situ magnetic field data. The bases of these models are in situ measurements with more than 28,000 vector field averages. With these data sets and the mathematical descriptions of the current system it is possible to describe the geomagnetic field.

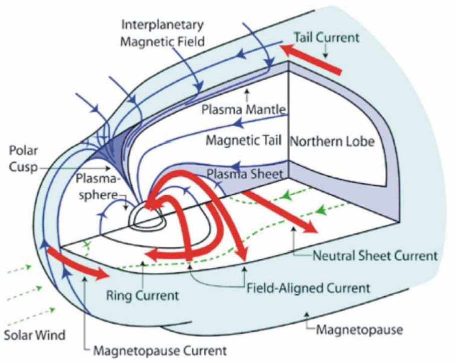


Figure 3.3: Magnetosphere (source: [44])

In order to describe the perturbations on the dipole field and the related influence of these currents in respect to the dipole field it is necessary to have a closer look at the currents which are present in the different regions of the magnetosphere (see fig. 3.3). The total magnetic field which is produced by extraterrestrial source and deforms the dipole field can be described as the combination of the magnetic field produced by these different currents namely the ring current, the magnetic tail current and the magnetopause current. With the mathematical description of the current systems [46, 49] and the adjustments due to the in situ measurements, Tsyganenko developed a very precise model of the geomagnetic field. The outer boundary of these model is defined as a paraboloid shaped cavity of the dipole field due to the effect of the solar wind pressure. The magnetic field $\vec{B}$ expected at a given time and location in space can then be predicted with this model. The field can be expressed by ($\vec{B} = -\vec{\nabla} V$) as

$$\begin{aligned} B_r &= \sum_{n=1}^{N} (n+1)\left(R_E/r\right)^{n+2} \sum_{m=0}^{n} \left(g_n^m \cos(m\phi) + h_n^m \sin(m\phi)\right) \\ &\quad \vec{P}_n^m \cos(\theta) \qquad (3.3) \\ B_\theta &= -\sum_{n=1}^{N} \left(R_E/r\right)^{n+2} \sum_{m=0}^{n} \left(g_n^m \cos(m\phi) + h_n^m \sin(m\phi)\right) \\ &\quad \vec{P}_n^m \cos(\theta)/\partial\theta \qquad (3.4) \\ B_\phi &= \sum_{n=1}^{N} \left(R_E/r\right)^{n+2} \sum_{m=0}^{n} m\left(g_n^m \sin(m\phi) - h_n^m \cos(m\phi)\right) \\ &\quad \vec{P}_n^m \cos(\theta)/\sin(\theta) \qquad (3.5) \end{aligned}$$

where r, θ, ϕ are the geocentric spherical polar coordinates. A visualisation of the field under different outer conditions is shown in fig. 3.4.

It can be seen that the field varies in time due to the effect of the solar wind and the position of the Earth. Therefore when a satellite is in orbit around the Earth the magnetic conjunction between the satellite and the ground varies also with time and with the outer conditions. In fig. 3.4 the satellite is shown in a typically distance of about 6.62 R_E. Depending on time and tilt of the Earth's axis the conjunction is different. In order to calculate the magnetic conjunction of the satellite in respect to the magnetic field the field lines which crosses the satellite path have to be traced back to the surface. This is done by the function `MagPOS` by using Tsyganenko's field line tracing algorithm which is described in the next section.

3.3 Field Line Tracing

The main field of application for the `GEOPACK` in this thesis is the numeric computation of the field lines and how they could be traced. The `GEOPACK` subroutine `TRACE` traces a filed line from an arbitrary point of space to the Earth's surface or to any specified boundary. Here the algorithm structure and the mathematical description of the tracing is presented. A field line of the general magnetic field which is described above could be expressed as follows

$$\frac{\partial r}{B_r} = \frac{r\partial\theta}{\partial B_\theta} = \frac{r\sin(\theta)\partial\phi}{B_\phi} = \frac{\partial s}{B} \tag{3.6}$$

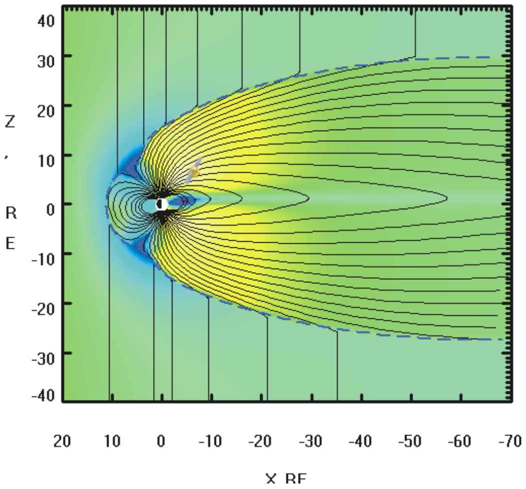
where ∂s denotes an element of arc length along a magnetic field line and $B = \left(B_r^2 + B_\theta^2 + B_\phi^2\right)^{1/2}$. The field $\vec{B}$ here is described as $-\vec{\nabla}V$ (cf. equation (3.2)). For the axis symmetric part the internal geomagnetic field can be rewritten

$$\frac{\partial r}{B_r} = \frac{r\partial\theta}{\partial B_\theta} \tag{3.7}$$

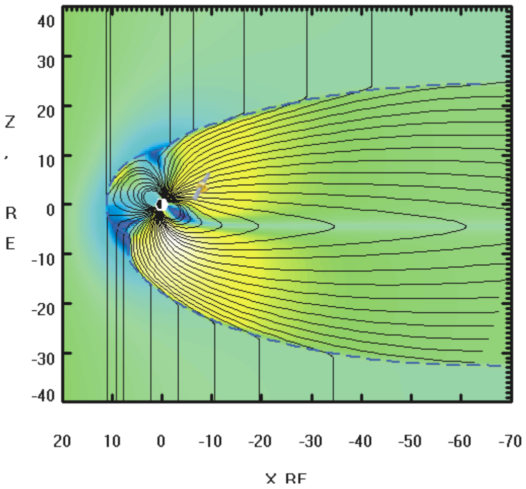
here $B_\phi = 0$. To trace along a magnetic field line, the subroutine performs a numerical calculation based on the conditions that the tangent at each point of the field line is parallel to the magnetic field at the same point. This fact is stated in eq. (3.6) and can be rewritten in vector form as

$$\partial\vec{r}/\partial s = \vec{B}/|B| = \vec{b} \tag{3.8}$$

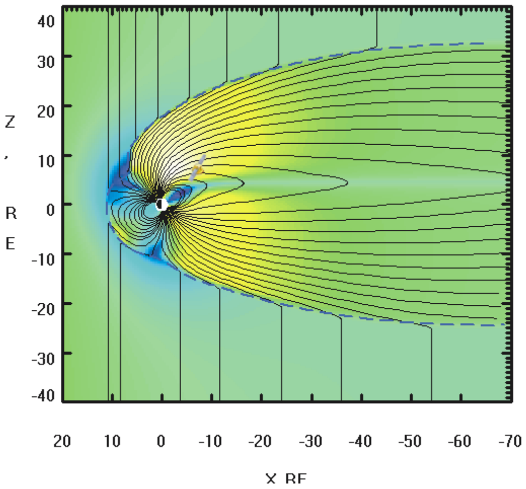
where $\vec{r}$ is the position vector measured from the center of the Earth, s is the arc length along the magnetic field line, $\vec{B}$ is the magnetic field vector and $\vec{b}$ is an unit vector parallel to the magnetic field line. With this expression it is possible to determine the next point on the field line by $\vec{r} + \partial\vec{r}$ the new position is given and the process is repeated until the boundary condition is reached. In order to estimate the stepsize $\partial\vec{r}$, $\vec{b}$ and ∂s must be known. The vector $\vec{b}$ is defined by $\vec{B}/|B|$, here the Tsyganenko model used the extended IGRF model to estimate the magnetic field. The value ∂s requires some assumptions about the shape of the field line, as a first assumption ∂s could be assumed as a linear connection but more accurate is to consider also the curvature of the field. In the subroutine it is assumed that the field line is dipolar between certain points, so that the field line traced is a succession of



(a) Earth's axis with zero tilt



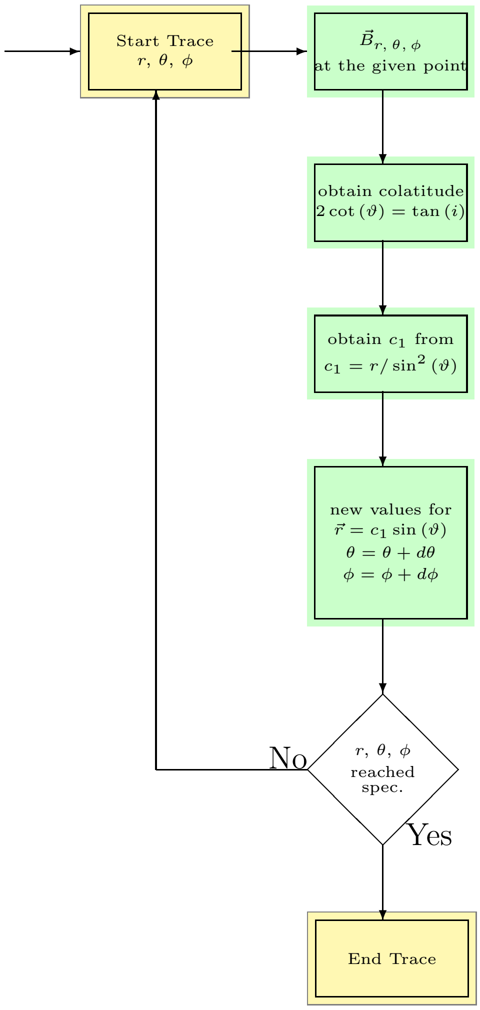
(b) Earth's axis with upwards tilt



(c) Earth's axis with downwards tilt

Figure 3.4: Tsyganenko's Geomagnetic Field Model (source: NASA/GODDARD Space Flight Center)

dipolar elements of arc length ∂s. The task of the subroutine then is to trace the field line along these small paths.



This is accomplished by determining first the location of the given point with respect to the dipole axis by calculating the corresponding dipole colatitude φ from the relation $2\cot(\vartheta) = \tan(i)$ where i is the inclination of the dipole field. With φ known the position of the next point is found by the change in φ that corresponds to ∂s. The increment can be calculated by

$$d\varphi = \frac{r}{R_E}\left(\frac{2\sin(\varphi)}{\sqrt{3\cos^2(\varphi)+1}}\right)\frac{\partial s}{R_E}, \tag{3.9}$$

where r is the geocentric distance of the point. A value of ∂s is required but this remains fixed during the trace. With these parameters it is possible to trace the field line step by step but for every step the values for r, θ, ϕ are changing and have to keep updated for the $\vec{B}_{r,\theta,\phi}$ calculation. Therefore the following equations have to be processed

$$r = c_1 \sin^2(\varphi + d\varphi) \tag{3.10}$$

$$\theta = \theta + \cos(\psi) d\varphi \tag{3.11}$$

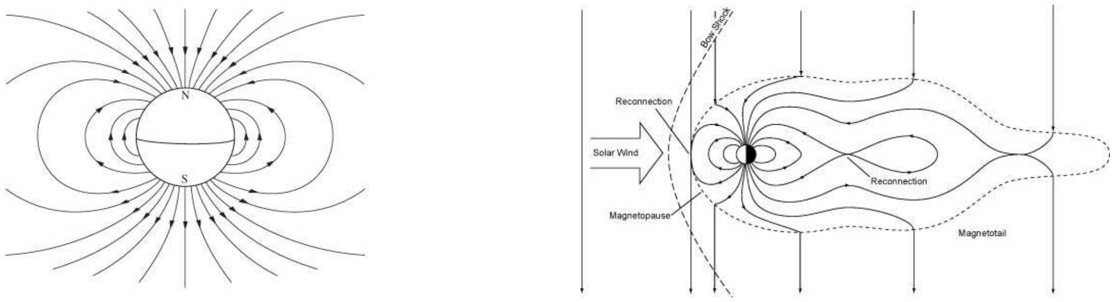
$$\phi = \frac{\sin(\psi)}{\sin(\theta)} d\varphi \tag{3.12}$$

where ψ is the magnetic declination. The structure diagram of these algorithm is shown on page 32. With these tracing it is possible to calculate the magnetic conjunction between a satellite and the ground station and to produce magnetic footprints. It should also be mentioned that the assumptions made in this section and by the model itselfs leads to uncertainties in the tracing in comparison to the measured points in space of about 5-10% see [39].

3.4 Summary

The Earth's magnetic field is both expansive and complicated. It is generated by electric currents that are deep within the Earth and high above the surface. The main field is caused by the geodynamo inside the Earth's core the hot and fluid iron in the core reacts due to its circulation like a self inducted dynamo and produces in combination with the currents above the surface due to charged particle motions, the magnetic field. The magnetic field of the Earth is often times described as being approximately dipolar, with field lines emanating from the south geomagnetic pole and converging at the north geomagnetic pole (see fig. 3.5 (a)).

The shape of the geomagnetic field then is influenced mainly by the Sun. The Sun generated its own tangled magnetic field that extends out into interplanetary space, also the Sun emits a wind of charged particles, a plasma that flows outwards into space and which carries with it the heliomagnetic field. These effects interacts with the Earth's magnetic field and deform the dipole structure. Because of the pressure exerted by the solar wind on the geomagnetic field, the field is compressed on the day side and elongated on the night side of the Earth (see fig. 3.5 (b)).



(a) dipolar character

(b) stretched dipole field due to the effect of the solar wind

Figure 3.5: Simple Shapes of the Earth's magnetic field (source: U.S. Geological Survey)

This shape is highly variable and far from stable, so the only way of describing such a field is by numerical computer models, which extrapolate the shape from magnetic measurements. Such a model is the IGRF model or as extension the Tsyganenko model which is used here. With such mean mathematical models it is possible to describe the shape of the field during solar events or to propagate the position of magnetic field lines. This is used in this thesis, the `GEOPACK` routines from Tsyganenko enables the user to simulate the geomagnetic field. So it is possible to calculate the magnetic conjunction between a satellite in a specified orbit around the planet and its magnetic conjunction. This is implemented in the function `MagPOS`.

4 Program/Toolbox (VIS SAT)

The first sections dealt with the theory behind the motion of satellites in the near Earth space and the description of the environment they are exposed to.

The motions could be described by Newton's laws of motion and the Keplerian orbit elements. The mathematical expressions should now be used to develop a program/toolbox to visualise the trajectories of satellites in space. The Tsyganenko model (see section 3.2) of the magnetic field is used to present the conjunction between the satellites and the Earth magnetic field by producing latitude and longitude of the magnetic footprints in the northern and southern hemisphere respectively.

In addition to the visualisation, the functions in **VIS SAT** have also the functionality to be used in a command line way to enable the user to involve the functions in further studies (e.g. communication links or space weather studies). **VIS SAT** is meant to be an open-source satellite tracking and orbital prediction program written under MATLAB® and C/C++[4] for scientific purpose to enable the user to study satellite/ground conjunctions. Users may redistribute it and/or modify it under the terms of the GNU General Public License. The incorporated models (SGP4/SDP4/TSY'03) underlie comparable open-source licenses.

The main **VIS SAT** program combines the functions `SatPOS` for the satellite position over ground (see section 4.1.3), `SatTRACK` tracking of the satellite position (see section 4.1.4) and `MagPOS` for conjunctions between the satellite and the magnetic field (see section 4.1.5) in a graphical user interface (GUI). The GUI uses the functionality of MATLAB® to produce three dimensional plots of the position and the orbit and two dimensional plots for the footprints. Also in addition to this the GUI has an external input tool to enable the user to specify Keplerian elements by hand. This tool converts the osculating classical Keplerian orbital elements to the standard TLE format to process the calculations with these data.

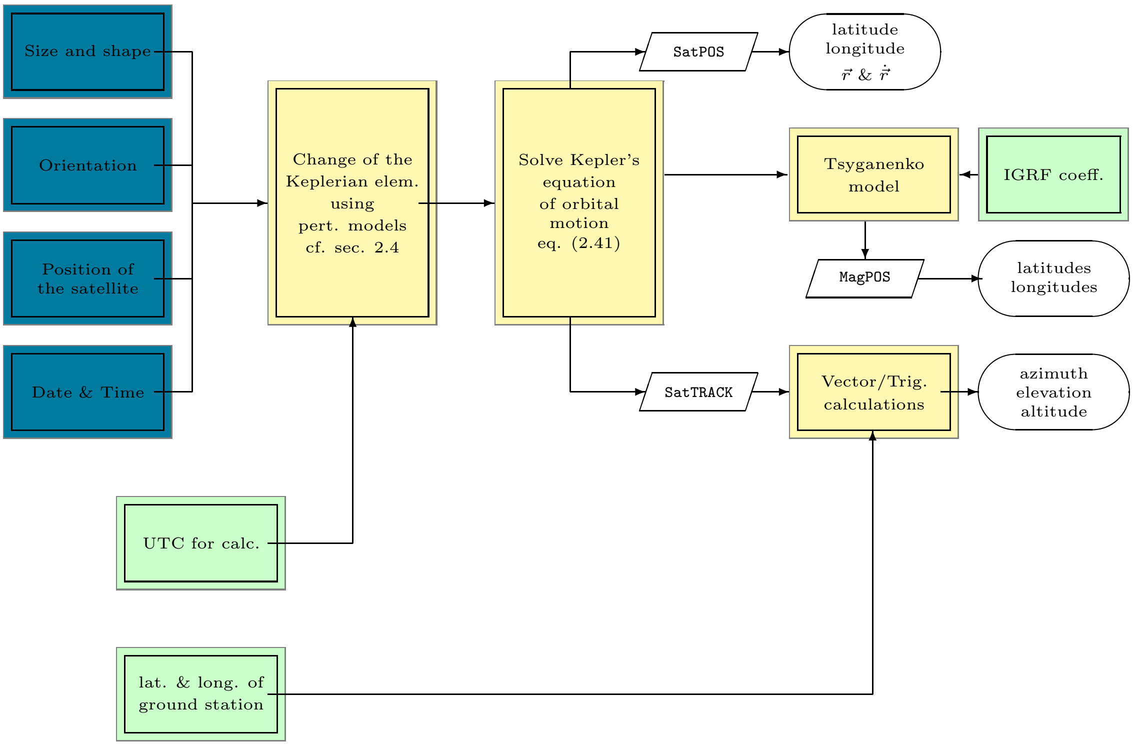
The contained functions are downwards compatible with MATLAB® 5.3 Release 12, for the GUI MATLAB® 7 Release 14 is required.

4.1 Structure and Implementation

This section deals with the overall structure of **VIS SAT** and how the models described in section 2 and 3 are fitted into this structure. The mathematical models described in the sections above represents the mathematical theory for the numerical calculations. Therefore the motion of a satellite is described by the SGP4/SDP4 models and the magnetic conjunction is described by Tsyganenko's model. The schematic on page 35 shows how the input data are processed during the calculations and which additional input data are necessary.

---

[4] here Microsoft Visual C++ 6.0 was used, in order to be able to convert the C/C++ programs to a `MEX` function, the function is written in a not object orientated way, this speeds up the calculation time


Size and shape
Orientation
Position of the satellite
Date & Time
Change of the Keplerian elem. using pert. models cf. sec. 2.4
Solve Kepler's equation of orbital motion eq. (2.41)
SatPOS
latitude longitude $\vec{r}$ & $\dot{\vec{r}}$
Tsyganenko model
IGRF coeff.
MagPOS
latitudes longitudes
SatTRACK
Vector/Trig. calculations
azimuth elevation altitude
UTC for calc.
lat. & long. of ground station

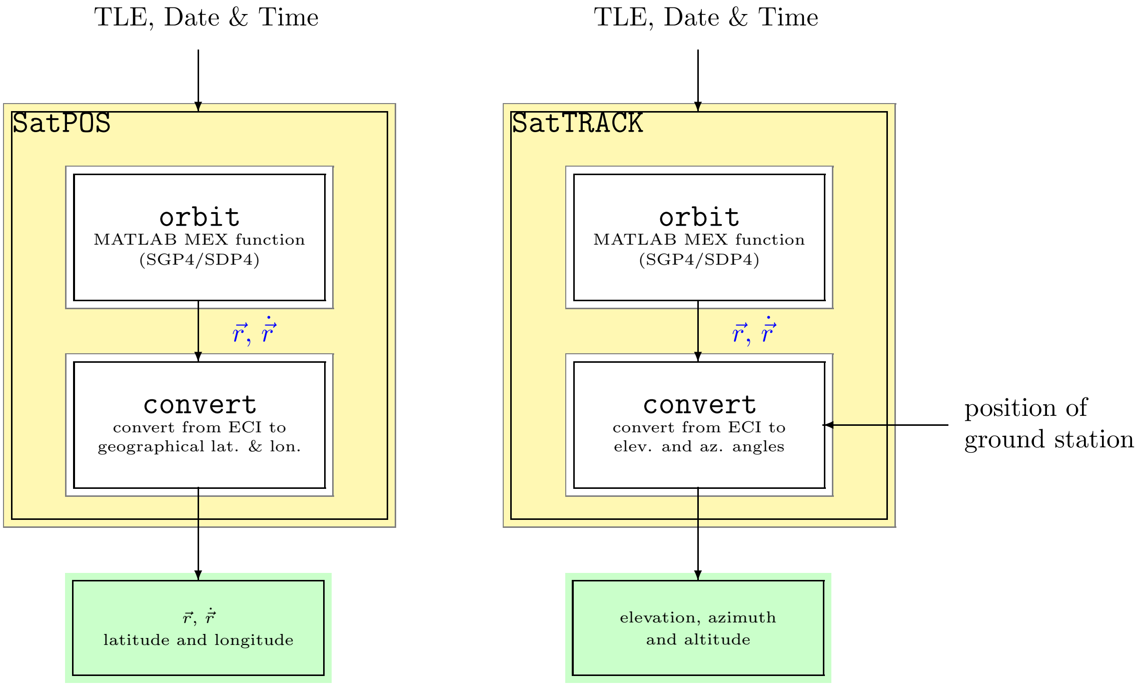
It can be seen that the original input data can be obtained from the NORAD TLE data sets (cf. section 2.4.1) which contains the Keplerian elements to determine the orbit. These data have to be processed according to the SGP4/SDP4 model specified by NORAD. So it is possible to calculate the Keplerian elements with respect to the perturbations and according to the flow chart on page 17. The position vector $\vec{r}$ in an ECI system and the velocity $\dot{\vec{r}}$ can then be calculated by using Kepler's equation for any given time. The function `SatPOS` then calculates with $\vec{r}$ the subsatellite point (SSP), the surface location on the Earth between the satellite and the geocenter, in geographical latitude and longitude. In addition to the SSP the functions also returns the position and velocity vector for further analysis. The function `SatTRACK` calculates the elevation and azimuth angle between a given ground station position and the satellite, in order to calculate when the satellite of interest will rise over the horizon and the appropriate azimuth angle at which the receiving antenna or dish must be pointed. The function `MagPOS` calculates the magnetic footprint in latitude and longitude of the satellite in the northern and southern hemisphere respectively.

Also some special design and implementation patterns should be mentioned here. The flow chart above shows the general structure of the calculation process. The change of the Keplerian elements due to perturbations can be calculated by the SGP4/SDP4 model and has to be done for all three functions. In order to provide maximum flexibility and to be in line with the guidelines from the GNU General Public License all three functions contains the MATLAB® `MEX` function `orbit`. This function is an `MEX` implementation of the SGP4/SDP4 models which are implemented in C/C++ to obtain a minimised computation time (here the SDP4 model was corrected by using [5, 29]). The compiler to create the `MEX` function was set on the `gcc`-compiler from Cygwin[5]. The function `orbit` is contained in all functions to

---

[5] Cygwin is an UNIX environment for Windows© which is distributed by RedHat. The main component is a dynamic link library (`dll`) with basic UNIX functions and to use these function in a Windows© environment (www.cygwin.com)

ensure a maximum of flexibility. This design enlarges the scope of applications.



It can be seen that the `MEX` (see appendix B.1) function `orbit` calculates the actual position $\vec{r}$ of a satellite and the velocity $\dot{\vec{r}}$, with these information and different transformation in respect to the aspired frame of reference, the functions calculate the satellite dynamic.

4.1.1 Read NORAD TLE

In order to provide the functions with the initial data and to perform the different transformations, **Vis Sat** contains an auxiliary function which performs this task. **Vis Sat** contains implementations of the SGP4 and SDP4 model in the `MEX` function `orbit`, in order to provide this function with the data from the NORAD TLE data sets the data have to be extracted from the files. The standard form of distribution of the TLE data is in form of `txt` files (due to new availability restrictions for TLE data **Vis Sat** contains several TLE data sets). The data structure is described in section 2.4.1 and can be read into MATLAB$^{\circledR}$ and stored in an array of struct. The function `norad2kep` (see appendix B.2) reads a TLE data set from a `txt` file using the `fgetL` function to read line by line. The call syntax of the function is shown below

```
>> satellite=norad2kep(file,name)
```

where `file` is the filename with suffix and `name` is the name of the satellite both as strings, the returning statement of the function is an array of struct.

```
satellite.name  [string]
satellite.tle   [1x11 double]
satellite.epoch [datevec, NORAD date]
```

The field `satellite.tle` contains the NORAD data to calculate the orbit position $\vec{r}$ and the velocity $\dot{\vec{r}}$ by using the SGP4/SDP4 model. These elements are mainly the Keplerian elements and the additional values to calculate the disturbances, here the first and second time derivations of the mean motion. The order of the elements in the field is shown in table 4.1 (the index 0 indicates that these values are the initial values read from the TLE data). This data structure is used in the main functions to pass the needed elements to the function `orbit`. The function `norad2kep` is a stand alone function, it contains a conversion from the standard TLE epoch format to the Julian date standard `tle(1)`. This function was validated by using the U.S. Naval Observatory Julian date converter (see http://aa.usno.navy.mil/data/docs/JulianDate.html). In addition to the Julian date, the array of struct contains also a field with a MATLAB® conform `datevec` with a structure of `[YYYY MM DD hh mm ss]` in an universal time (UT) frame and a field which contains the original NORAD set `yyddd.fffffff`. This provides a wide range of flexibility just as time calculations with software tools are often difficult to handle.

Position	Math. Symbol	Meaning
`tle(1)`	t_0	epoch as Julian date
`tle(2)`	$\dot{n}_0$	1st derivation of mean motion
`tle(3)`	$\ddot{n}_0$	2nd derivation of mean motion
`tle(4)`	B^*	ballistic coefficient
`tle(5)`	i_0	inclination
`tle(6)`	Ω_0	right ascension of the ascending node
`tle(7)`	e_0	eccentricity
`tle(8)`	ω_0	argument of perigee
`tle(9)`	M_0	mean anomaly
`tle(10)`	n_0	mean motion
`tle(11)`	rev	revolution number at epoch

Table 4.1: Array structure of TLE

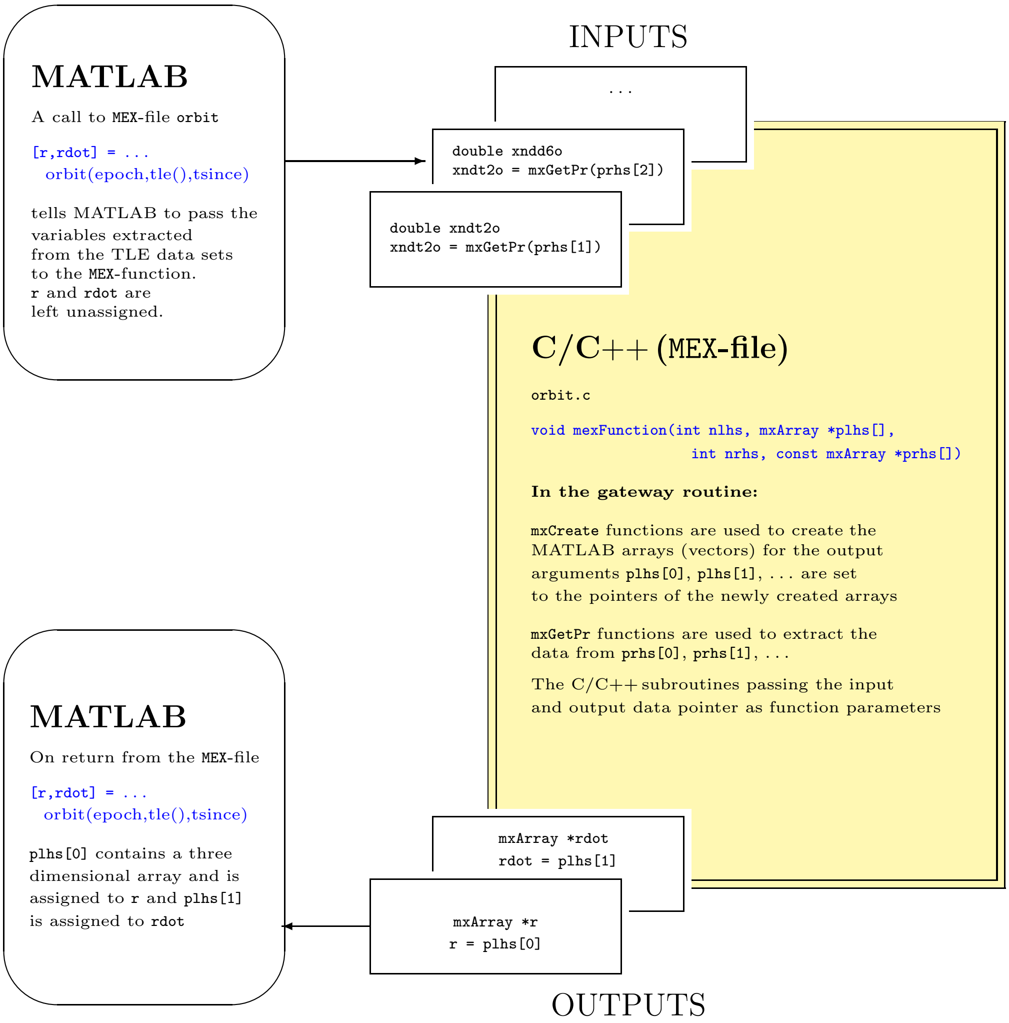
4.1.2 SGP4 and SDP4 implementation (`MEX` Function `orbit`)

The `MEX` implementation of the SGP4 and SDP4 model in the function `orbit` is rudimentary for the propagation process[6]. This function allow it to calculate the position $\vec{r}$ and velocity $\dot{\vec{r}}$ of a satellite at any given time by using the TLE data (cf. section 2.4.2). The implementation is done in two steps. The first step is to convert the models which are specified in the Spacetrack Report No. 3 [20] from the original FORTRAN code to C/C++ , in respect to the corrections made during the years, especially in the SDP4 subroutine `DEEP` (see appendix A.4.3). The implementation is a straight forward implementation, to ensure full compatibility with further applications, the C/C++ code contains the same variable names and structures as in the original FORTRAN routines, this ensures that further investigations or changes in the models in newer Spacetrack Reports could be implemented in the

---

[6] the `orbit.dll` produced from the C/C++ code is executable under Windows©, for a LINUX/UNIX system the `mex`-compiler of this system has to be used `>> mex [options] orbit.c`

C/C++ code without making great changes. The conversion from FORTRAN to C/C++ is chosen because of some incompatibilities (see also section 5 future work) of the FORTRAN/`MEX` interface. This conversion to C/C++ is necessary for the second step, a MATLAB® `MEX` function as a wrapper around the C/C++ code in order to create an interface to MATLAB®. The interface structure and the data flow is shown in the schematic on page 38.



A call to `MEX`-file `orbit`

```
[r,rdot] = ...
  orbit(epoch,tle(),tsince)
```

tells MATLAB to pass the variables extracted from the TLE data sets to the `MEX`-function. `r` and `rdot` are left unassigned.

```
double xndd6o
xndt2o = mxGetPr(prhs[2])
```

```
double xndt2o
xndt2o = mxGetPr(prhs[1])
```

`orbit.c`

```
void mexFunction(int nlhs, mxArray *plhs[],
                       int nrhs, const mxArray *prhs[])
```

In the gateway routine:

`mxCreate` functions are used to create the MATLAB arrays (vectors) for the output arguments `plhs[0]`, `plhs[1]`, . . . are set to the pointers of the newly created arrays

`mxGetPr` functions are used to extract the data from `prhs[0]`, `prhs[1]`, . . .

The C/C++ subroutines passing the input and output data pointer as function parameters

On return from the `MEX`-file

```
[r,rdot] = ...
  orbit(epoch,tle(),tsince)
```

`plhs[0]` contains a three dimensional array and is assigned to `r` and `plhs[1]` is assigned to `rdot`

```
mxArray *rdot
rdot = plhs[1]
```

```
mxArray *r
r = plhs[0]
```

A detailed description of the the `MEX` function syntax can be seen in appendix B.1 the realisation in the function `orbit` can be seen in appendix B.3. The TLE parameters are passed into the `prhs` pointer and assigned to the `tle` structure in the C/C++ file. The C/C++ code contains a structure for the TLE elements like the structure in `norad2kep` to provide the data for the orbit calculations.

```
/* Two-line-element satellite orbital data
   structure used directly by the SGP4/SDP4 code. */
typedef struct {
   double  epoch, xndt2o, xndd6o, bstar, xincl,
          xnodeo, eo, omegao, xmo, xno;
  int      revnum;
 } tle_t;
```

During the `MEX` call, these variables are assigned with the elements read from the

TLE file (see schematic page 38). In the C/C++ code the original variables from the FORTRAN routines are used, therefore here the assignment structure from the function `norad2kep` to the C/C++ structure is shown in table 4.2. The variable `tsince` is not defined by the `norad2kep` but by the **Vis Sat** functions itself, the time is calculated by using the external given time (see dataflow on page 35) and the extracted epoch time.

MATLAB® field	C/C++ variable	Meaning
`sat.epochN`	`epoch`	epoch as NORAD date
`sat.tle(2)`	`xndt2o`	1st derivation of mean motion
`sat.tle(3)`	`xndd6o`	2nd derivation of mean motion
`sat.tle(4)`	`bstar`	ballistic coefficient
`sat.tle(5)`	`xincl`	inclination
`sat.tle(6)`	`xnodeo`	right ascension of the ascending node
`sat.tle(7)`	`eo`	eccentricity
`sat.tle(8)`	`omegao`	argument of perigee
`sat.tle(9)`	`xmo`	mean anomaly
`sat.tle(10)`	`xno`	mean motion
`sat.tle(11)`	`revnum`	revolution number at epoch
`tsince`	`tsince`	time since epoch

Table 4.2: Assignment Structure between MATLAB® and C/C++

The main body of the function `orbit` contains the two following *"main"* functions

```
SGP4(double tsince, tle_t * tle, double *position, double *velocity)
```

and

```
SDP4(double tsince, tle_t * tle, double *position, double *velocity)
```

this functions implement the models described in section 2.4.2. The variable `tsince` is the time since epoch in minutes, `tle` is a pointer to a `tle_t` structure with Keplerian orbital elements and `position` and `velocity` are three dimensional vector structures returning ECI satellite position and velocity. These two vectors are the left hand side values of the `mexFunction` and returned to MATLAB®.

In the original Spacetrack Report No. 3 [20] it was mentioned that the model should be used according to the orbit time period, SGP4 for under 225 minutes and SDP4 for orbit periods over 225 minutes, it can be calculated by equation (2.29). Nevertheless the Spacetrack Report No. 6 stated, that this definition in combination with the SDP4 model is not quiet accurate enough, because this definition does not take into account the other orbital elements. Therefore a new empirically definition is used (here the C/C++ implementation is shown)

```
if (twopi/xnodp/xmnpda>=0.15625)
   SetFlag(DEEP_SPACE_EPHEM_FLAG);
 else
   ClearFlag(DEEP_SPACE_EPHEM_FLAG);
```

where `DEEP_SPACE_EPHEM_FLAG` is a flag which indicates the SDP4 model has to be used. According to the `if` clause it is set when the condition is true or cleared otherwise. The condition can be rewritten by using orbital parameters in the following way

$$\frac{2\pi}{1440}\left(\frac{c_1 n_0^{1/2}\left(3\cos^2\left(i_0\right)-1\right)}{c_2^{3/2}a_0^2}+\frac{1}{n_0}\right)\geq 0.15625 \tag{4.1}$$

where c_1 and c_2 are empirically constants. The 1440 indicates the minutes per day. Using this `if` clause the distinction between the two models is more accurate. The C/C++ code of the implementation is shown in appendix B.3. All `GO TO` statements in the original FORTRAN code are replaced by the use of `FLAG`s. The function `orbit` is a stand alone function and can be used for different orbit propagations, the input and output structure is documented in `orbit.m`, nevertheless it should be mentioned here, that the C/C++ code has no own exception handler, in respect to the `MEX`-handler here it was forbear to using own exception handler. Therefore the function can only be used in direct assignment of all input statements and it must be insured that all input variables are correct. In the **Vis Sat** structure the orbit function is encapsulated in the functions `SatPOS`, `SatTRACK` and `MagPOS` these function contain the normal JAVA exception handler and control the input of errors by using JAVA `msgbox(...)` to control the data flow. This encapsulation is described in the next section.

4.1.3 Satellite Position Footprint (`SatPOS`)

The function `SatPOS` calculates the subsatellite point in geographical latitude and longitude. It encapsulates the function `orbit` and controlls the input and output data of the orbit propagation. The input variables are an array of struct, which contains the satellite TLE data, read in by the `norad2kep` function, and the actual time for which the SSP has to be calculated. This date and time must be given in a MATLAB® `datevec` standard to calculate `tsince` the minutes between the epoch and the given time. This is done by

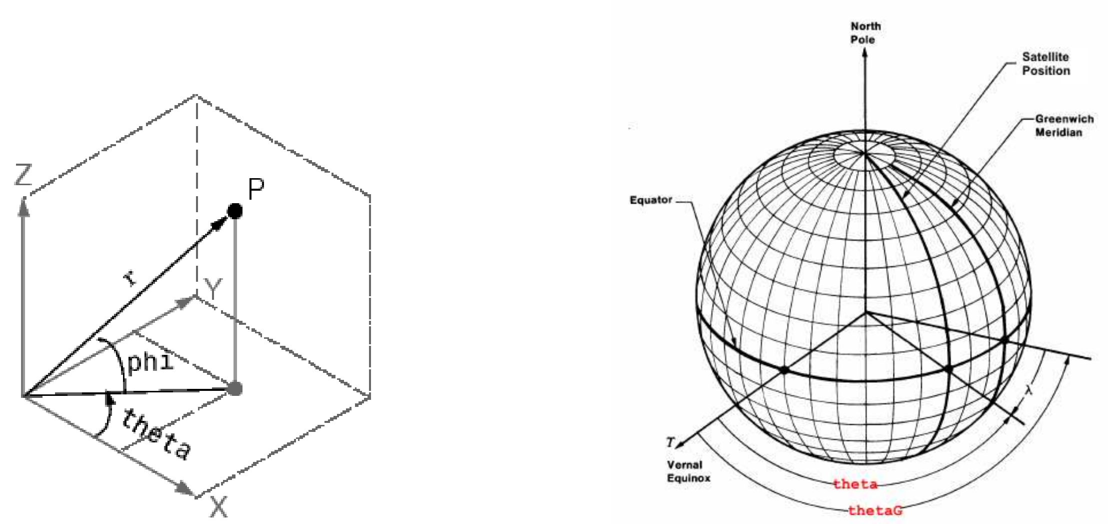
```
% time since epoch in minutes
tsince=(jd-sat.tle(1)).*1440
```

where `jd` is the given time in a Julian date and `sat.tle(1)` is the epoch time in a Julian date format. The time conversion of the given `datevec` to a Julian date format is done in a subfunction of `SatPOS`.

This time difference and the Keplerian orbit elements recovered from the array of struct is passed to the `orbit` function to calculate the position and velocity of the satellite due to the time since epoch. The function call is the follows

```
[r,rdot]=orbit(sat.epochN,sat.tle(2),sat.tle(3),sat.tle(4),sat.tle(5),...
                  sat.tle(6),sat.tle(7),sat.tle(8),sat.tle(9),...
                     sat.tle(10),sat.tle(11),tsince);
```

where the `sat.tle(...)` structure is passed to the `MEX` function and assigned in this function in accordance with table 4.2. The SGP4/SDP4 implementation returns the three dimensional position vector in `r` ($\vec{r}$) and the velocity in `rdot` ($\dot{\vec{r}}$). Then with the position of the satellite in an ECI reference frame the latitude and longitude can be calculated. Therefore the ECI state vector has to be transformed to a geodetic system. This is done by converting the ECI coordinates into spherical coordinates in respect to the Earth's latitude and longitude graduation, where the latitude is from -90° to 90° with the equator as zero degrees and the longitudes from 0° to 360° with the Greenwich meridian as zero degrees. This is shown in fig. 4.1.



(a) general cartesian to spherical coordinate transformation (source: MATLAB®)

(b) taken into account the Greenwich Mean Sidereal Time (GMST)

Figure 4.1: ECI to Geodetic Conversion

The azimuth `theta`/θ and the elevation `phi`/ϕ are angular displacements in radians measured from the positive x-axis, and the $x-y$ plane, respectively and `r` is the distance from the origin to the satellite's position. These values can be calculated by

$$\texttt{theta} = \text{actan2}\,(y, x) \tag{4.2}$$

$$\texttt{phi} = \text{actan2}\left(z, \sqrt{x^2+y^2}\right) \tag{4.3}$$

$$\texttt{r} = \sqrt{x^2+y^2+z^2} \tag{4.4}$$

where x, y and z are the position elements of the satellite's ECI vector and the actan2 is the four-quadrant arcus tangents (this is shown in fig. 4.1 (a)). The x-axis of the ECI system is pointed to the vernal equinox (cf. section 2.1.1) to transform the ECI system to the geodetic system the GMST `thetaG`/θ_G has to be taken into account (see fig. 4.1 (b)). The GMST can be calculated as follows

$$\begin{aligned}\texttt{thetaG} = {} & 280.46061837 + 360.98564736629\,(JD - 2451545.0) \\ & +0.000387933T^2 - T^3/38710000\end{aligned} \tag{4.5}$$

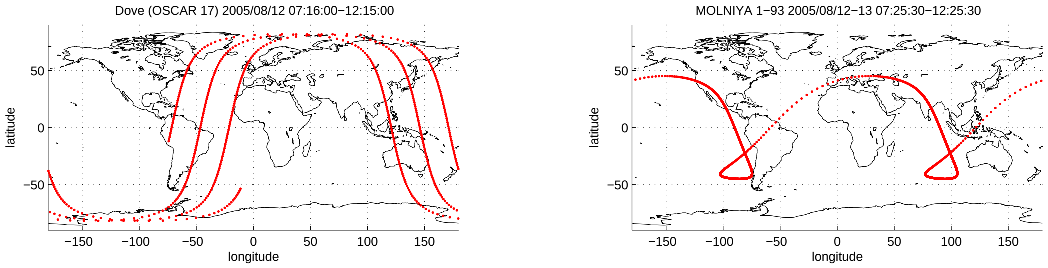
where $T = (JD - 2451545)\,/36525$ and JD is the Julian date. This calculation was specified by the International Astronomical Union (IAU) in 1980. With this GMST hour angle in radians it is possible to correct `theta`/θ and to correct the longitude by this factor. The latitude is given by `phi`/ϕ but has to be corrected due to the effect that the Earth is not a sphere, therefore a flattening factor f is introduced. To calculate the subsatellite point by approximating `phi`/ϕ with `phi`'/ϕ' and loop to the following calculations specified by IAU

$$\phi = \phi_i \tag{4.6}$$

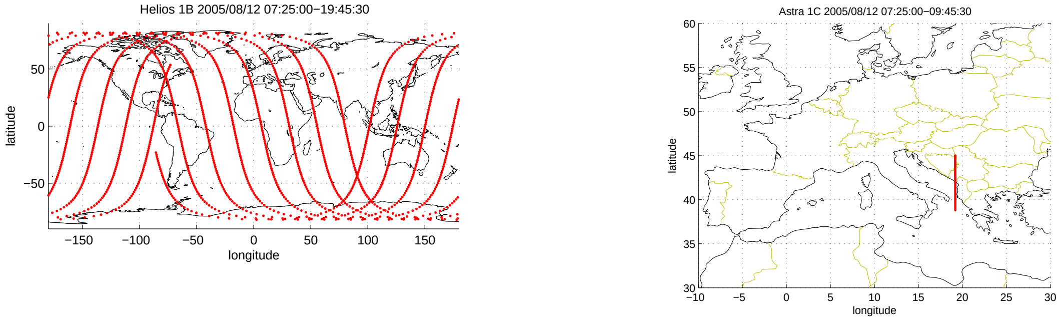
$$c = \frac{1}{\sqrt{1 - e^2 \sin^2(\phi_i)}} \tag{4.7}$$

$$\phi = \arctan\left(z + c a_E e^2 \sin^2(\phi_i)\right) \tag{4.8}$$

until $|\phi - \phi_i|$ is within the desired tolerance, here `1e-10`. The implementation is shown in appendix B.4, the computation time analysis (accomplished with the in-built `tic` and `toc` command) shows that the complete function `SatPOS` needs less than 0.01 seconds to calculate the footprint of the satellite at one specified time. These results were achieved on a Windows© PC AMD XP-M1800+, with 512MB 400 MHz DDR-RAM in MATLAB® Version 6.5 (Release 13).


Dove (OSCAR 17) 2005/08/12 07:16:00−12:15:00
latitude
50
0
−50
−150
−100
−50
0
50
100
150
longitude
MOLNIYA 1−93 2005/08/12−13 07:25:30−12:25:30
latitude
50
0
−50
−150
−100
−50
0
50
100
150
longitude

(a) communication satellite (low Earth orbit) (b) molniya satellite (molniya orbit)



(c) european imagery satellite (polar orbit) (d) broadcasting satellite (geo. orbit)

Figure 4.2: Geodetic Position Footprints

The function was validated with the GODDARD Space Flight Center SSC Locator, the Locator component provides tabular information about the coordinate location of specified spacecrafts (see http://sscweb.gsfc.nasa.gov/). The validation shows that the implemented orbit calculation and ECI conversion corresponds with the high precision orbit calculation in the Locator up to the second decimal place and up to three month in advance simulation. The accuracy of the predictions is dependent upon several factors. Firstly, a satellite's orbit is affected by the tenuous upper atmosphere which creates a small, but continuous drag force. This drag is proportional to the density of the ionosphere, which is notoriously difficult to predict. Solar activity can increase the density by over a hundred times more than its average value. Secondly, because of this residual air drag, some active satellites reboost themselves to a higher orbit. These boost manoeuvres are also unpredictable in the mathematical models.

The orbit predictions are essentially calculated by taking the actual measured position and velocity of a satellite at a given point in time (the epoch), and propagating forwards in time taking account of the major forces, which are the Earth's gravitational attraction and the air drag. Given the uncertain nature of the air drag force acting on the satellite, the further the orbital position is predicted into the future, the more inaccurate it will become. This means that the latest orbital parameters should always be used. Because of this fact it is necessary to keep the TLE data (included in **Vis Sat** for Aug. 2005) updated at least every 3 month to achieve accurate positioning (the data sets can be downloaded from http://www.space-track.org).

TLE data from the 12th August 2005 have been used to calculate the different footprints shown in fig. 4.2.

It can be seen that every satellite has a characteristic orbit footprint which depends on its purpose. The footprint of `OSCAR 17` shows that the satellite is reachable most of the time in the latitude range from -50° to 50° for communication purposes and circles the whole Earth in about 90 minutes. The molniya orbits with a typical inclination of about 60° shows that the satellite is a long period reachable from Russia due to the fact of an apogee of over 38,000.00 km. The perigee lies by 1,000.00 km, therefore the satellite stays longer times in the apogee position and provides longer visibilities over Russia. The `Helios` satellite is a polar orbit satellite for imagery purposes, the satellite can survey the whole of the Earth's surface. The `Astra 1C` satellite is in a geostationary orbit for broadcast purposes. It is fixed on a longitude position of 19.2° east (see also http://www.ses-astra.com/satellites/-fleet/index.shtml). The latitude variation is due to the effect, that the orbit is not direct in the equatorial plane. These footprints should be representative for the functionality of `SatPOS` and shows how the data calculated could be processed. In addition to the two dimensional plots MATLAB® also provides three dimensional plots to visualise the position of the satellite in respect to the Earth. This is shown in fig. 4.3 and is an element of the **Vis Sat** GUI.

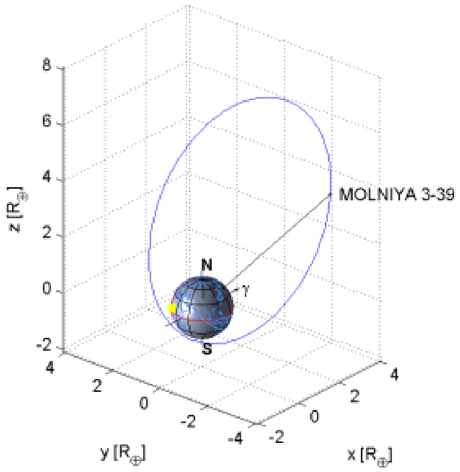


Figure 4.3: Three Dimensional Simulated Trajectory of Molniya 3-39 satellite with Earth satellite conjunction and orbit path

4.1.4 Ground Station Tracking (`SatTRACK`)

In addition to the geodetic position of the subsatellite point it is necessary to calculate the visibility of a satellite from a given point on the Earth's surface in a topocentric horizon coordinate system (see section 2.1.3). Therefore in accordance with the schematic on page 35 it can be seen that for these calculations the position of the ground station is needed in addition to the time of calculation. Due to the given time the position of the satellite can be calculated analogues to the calculations made in `SatPOS` (see section 4.1.3) with the position of the satellite and the given ground station position it is possible to calculate the azimuth and elevation of the satellite in respect to the ground station.

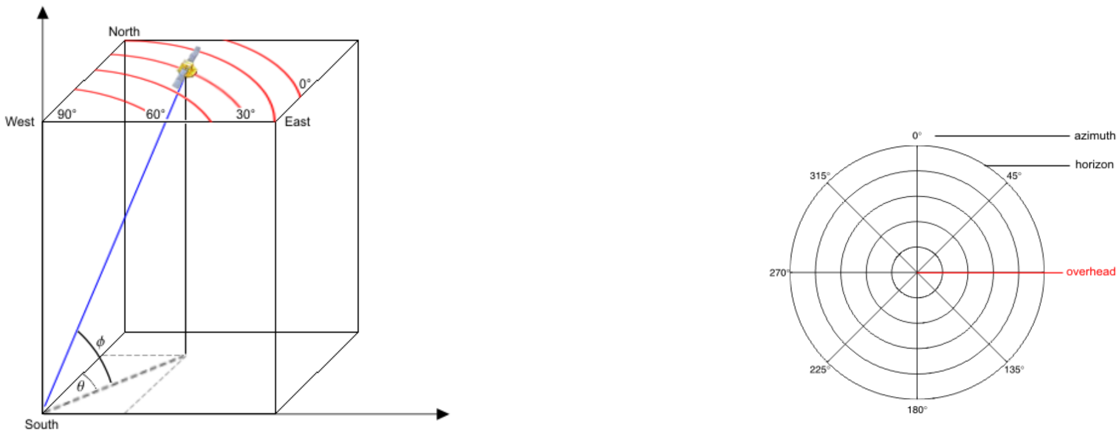
The input of the position of the ground station is conform to the European Space Agency (ESA) standard in latitude and longitude with decimal places. For latitudes positive numbers indicates degrees north and negative numbers south ($-90°$ - $+90°$), for the longitude positive values indicate west and negative east ($-180°$ - $+180°$). The ground station altitude is given in meters with respect to the World Geodetic System (WGS) reference ellipsoid. At the first step these geodetic coordinates have to be converted to an ECI frame. In general this transformation can be described as follows (see also [7])

$$\underline{r}^B = \underline{T}_A^B \left(\underline{r}^A - \underline{r}_A^B\right) \tag{4.9}$$

where $\underline{T}_A^B$ is the transformation matrix between the two different reference frames, here in general A and B and $\underline{r}_A^B$ is the origin of frame B with respect to frame A. According to this reference frame transformation the azimuth θ and the elevation ϕ to an object in the reference frame B are given by the following equations

$$r_x^B = \left|\underline{r}^B\right| \cos(\phi)\sin(\theta) \qquad r_y^B = \left|\underline{r}^B\right| \cos(\phi)\cos(\theta) \qquad r_z^B = \left|\underline{r}^B\right| \sin(\phi) \tag{4.10}$$

$$\theta = \text{actan2}\left(r_x^B, r_y^B\right) \qquad \phi = \arcsin\left(r_z^B / \left|\underline{r}^B\right|\right). \tag{4.11}$$

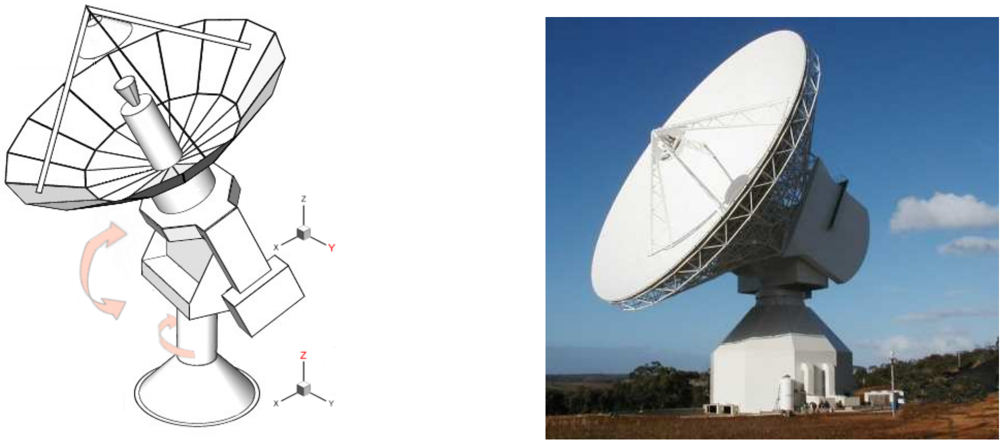


(a) skymap geometry (b) skymap plot

Figure 4.4: Skymap

So it is possible to calculate the elevation and azimuth angle for a given ground station in respect to the satellite's position at a given time. These angles can be used to align the dish of the ground station according to the path of the satellite above it to ensure a reliable up and down link connection. Figure 4.4(a) shows the geometry between the satellite and the ground station. To visualise these conjunctions with respect to the alignment angles, so called *"skymaps"* are used. These are polar plots which indicates the elevation and azimuth of an antenna see fig. 4.4(b).

The outermost circle in the plot represents the horizon 0° elevation. The center of the skyplot is 90° elevation, or directly overhead. The degree graduation represents the azimuth angle. Therefore north is at the top of the skyplot, east to the right, south at the bottom, and west to the left. With these plots it is possible to predict the visibility of a satellite over the local horizon, the calculation produces negative elevation angles when the satellite is under the local horizon and therefore not visible. In the skymap plots this could not be represented due to the fact, that the outermost circle represents a 0° elevation angle. This elevation and azimuth position determination of the satellite is also reflected in the general engineering design of ground stations for tracking satellites (see fig. 4.5).



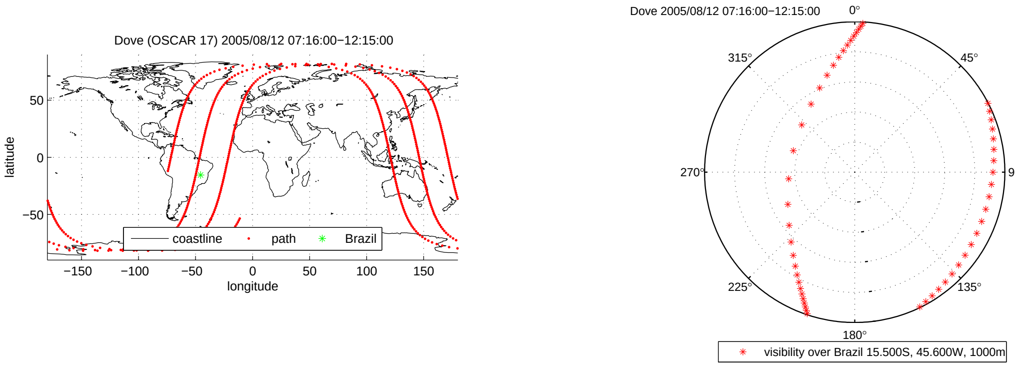
(a) schematic of a ground station with rotation axes

(b) ESA ground station in New Norcia, western australia (ESA ©)

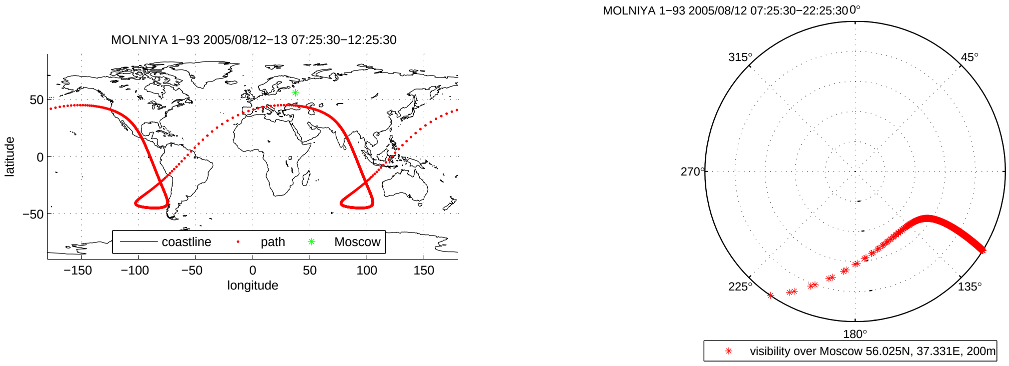
Figure 4.5: Ground Station

It can be seen that the stations have two rotation axes for their pan and tilt motion to track the satellite. The pan motion (the rotation about the z-axis) represents the azimuth position of the satellite. The tilt motion (the rotation about the y-axis) represents the elevation of the satellite position. This two degrees of freedom enables the ground station to follow the path of the satellite over the ground. The schematic in fig. 4.5(a) shows an ideal model of a ground station with a 360° mobility around the z-axis and a 180° mobility around the y-axis. This is often due to the real design (cf. fig. 4.5(b)) not possible, therefore the antenna elevation and azimuth limitations have to be taken into account when visualising the data from the function `SatTRACK` for a special device. Figure 4.6 shows three generalised skymap plots produced with the data calculated with `SatTRACK`. So the calculated data can be used for recalibration or tracking satellites over the ground.

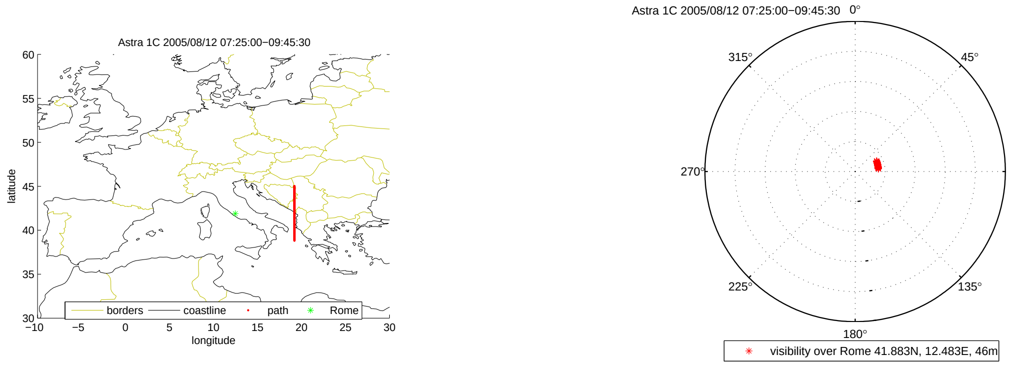
It should be mentioned here, that the function `SatTRACK` calculates the direct visibility of the satellite not the direct communication path. The calculations and plots here are only representative for communication conjunctions when the satellite has antennas directed to the Earth's center. Satellite could also have different squint



(a) communication satellite (b) oscar skymap



(c) molniya satellite (d) molniya skymap



(e) broadcasting satellite (f) broadcasting skymap

Figure 4.6: Skymap plots to track Satellites

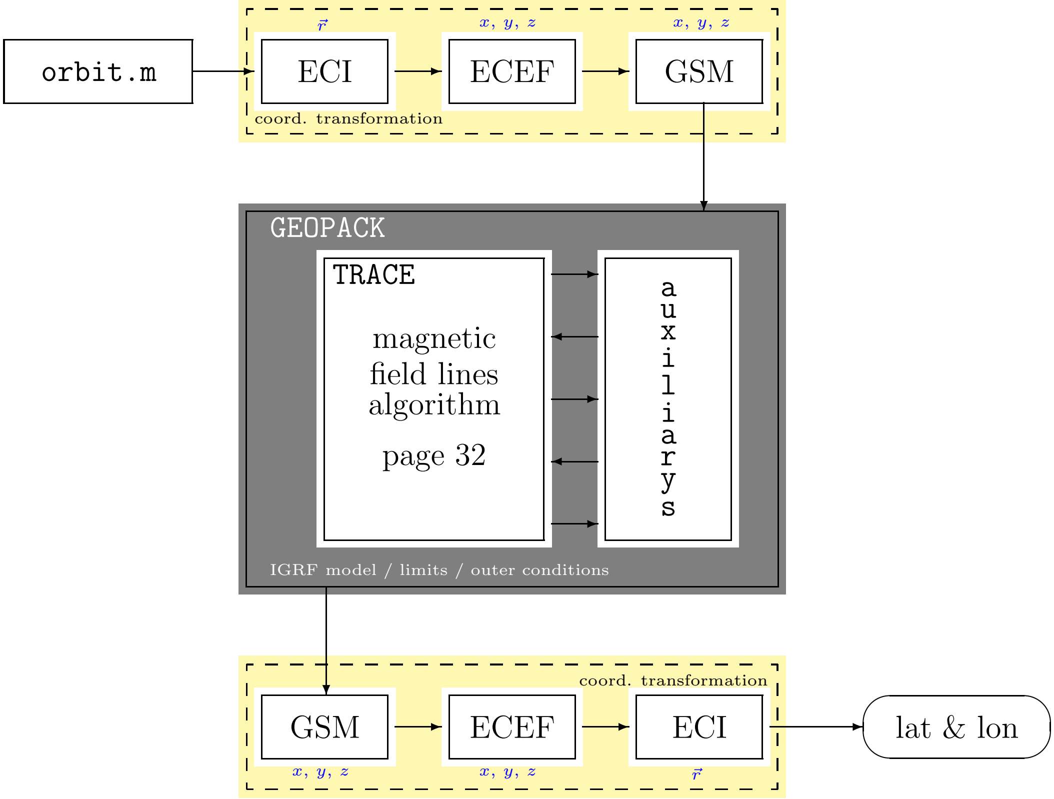
angles of their antennas to cover different areas, this squint angles could be obtained from several data bases and have to be taken into account by calculating the visibility criterion (see [31], http://www.satcom.co.uk/), especially for geostationary broadcast satellites these squint angles have to be taken into account for the elevation of the dishes.

The function `SatTRACK` and the aforementioned function `SatPOS` represents the flight dynamic part of the toolbox. These functions enables the user to calculate the geodetic position of the SSP or the satellite track over the local horizon. Both functions are tested and validated with GODDARD Space Flight Center data and both produce high accuracy propagations. The `SatTRACK` function is secondary tested by elevation and azimuth calculators for geostationary broadcast satellites. The tests with the BNN Communication System Engineering (see http://www.bnncom.com/satellite_azimuth_elevation.htm) shows good matches, nevertheless these broadcast calculations are for home satellite devices and not very accurate, they take just the mean values into account.

In addition to the flight dynamics and the motional conjunction, the toolbox also contains a tool to calculate the magnetic conjunction by using a numerical model of the Earth magnetic field (cf. section 3.2), the next section describes the implementation in MATLAB®.

4.1.5 Magnetic Footprint (`MagPOS`)

The sections above described how the satellite's flight dynamic footprints could be calculated and how they could be visualised. In addition to these satellite ground conjunctions **VIS SAT** also contains a function to study the magnetic conjunction of a satellite and the ground by considering the Earth's magnetic field (see fig. 3.4) and the position of the satellite in this field. Therefore the Tsyganenko model of the Earth magnetic field (revised Version 2003 based on the 1996 `GEOPACK`) is implemented in MATLAB® and the `GEOPACK` subroutines are used to trace fieldlines. The function `MagPOS` encapsulates the `GEOPACK` routines and represents the MATLAB® interface for the TSY'03 model and here all necessary coordinate transformations are processed. These transformations and the dataflow is shown in the schematic on page 47.



The `orbit MEX`-function provides the position vector $\vec{r}$ of the satellite in respect to the TLE data and the time since epoch. This position vector has to be preprocessed before passing to the `GEOPACK` routine `TRACE`. The vector is in an ECI reference frame and has to be transformed to a Geocentric Solar Magnetospheric system (GSM, the x-axis is defined from the Earth to the Sun, the y-axis is defined to be perpendicular to the Earth's magnetic dipole, so that the x-z plane contains the dipole axis) in order to start the calculations in the TSY'03 model. This is done in two steps, first the ECI vector elements are transformed to an ECEF frame of reference and then to the GSM elements. Then this vector is passed to the TSY'03 model where the field line is traced from the location of the satellite to an altitude of 100 km (default value could be changed) above the surface. The function `TRACE` traces the fieldline up to the given position above the surface. The output is also in an GSM reference frame. In order to calculate the latitude and longitude position of the footprint, the coordinate transformation is done reversed (cf. [42]). The algorithm used in the function `TRACE` to trace the fieldline is described in section 3.3, due to the fact that there are incompatibilities between the MATLAB® `MEX`-function for FORTRAN, here the direct implementation of the code is chosen which leads to a disadvantage in the computation time in comparison to a direct `MEX` implementation. The function `MagPOS` needs to calculate the northern and southern B-field trace between 2 and 10 seconds (measured with `tic` and `toc`). This problem is further discussed in section 5. The function needs as additional input also IGRF coefficients, K_p-values and $B_{y,z}$-field strengths respectively. The IGRF coefficients are calculated by the `GEOPACK_IGRF_GSM` the K_p and $B_{y,z}$ respectively could be given as a parameter during the call

```
[latN,lonN,latS,lonS]=MagPOS(ctime,sat,r0,parmod,iopt)
```

where `ctime` represents the UT for which the footprint should be calculated, `sat` represents the satellite data extracted from the TLE data set with the function `norad2kep`, `r0` represents the the given height above the surface to which the field line should be traced (default 100 km), `iopt` can be used for specifying an option of the external field model here by an interval of the K_p indexes (default dummy `parmod` is used) and `parmod` containing parameters needed for a specification of the external field. The vector contains the solar wind pressure in nPa (1st position), the disturbance storm time index (DST) in nT (2nd position), the B-field component in y-direction in nT (3rd position) and the B-field component in z-direction in nT (4th position). The default values are 3 nPa for the solar wind, -20 nT for the DST, 3 nT for the B_y component and -3 nT for the B_z component (for details see [11, 38, 44], the average values are based on [57]). The values can be set according to actual measurements and can be downloaded from the internet.

The function is validated with the SSC LOCATOR (used same default values) and shows minor differences in the B-field trace in the second decimal place.[7]. The longitude and latitude footprint of the satellite is given in `latN` and `lonN` for the

---

[7] due to an email from Paul O'Brien this is caused by the fact that the SSC Locator uses the Tsyganenko model in its 1989C version but with newer IGRF coefficients, the model implemented here is the 2003 updated version which is based on the Tsyganenko model version 1996. Due to the fact that neither measurements nor high precision orbit calculators could be such precise, here the comparison between the SSC Locator and the `MagPOS` calculations are considered as accurate enough

northern hemisphere and `latS` and `lonS` for the southern hemisphere, a footprint visualisation analogous to the SSP footprint is shown in fig. 4.7 (validation data can be seen in appendix B.7).

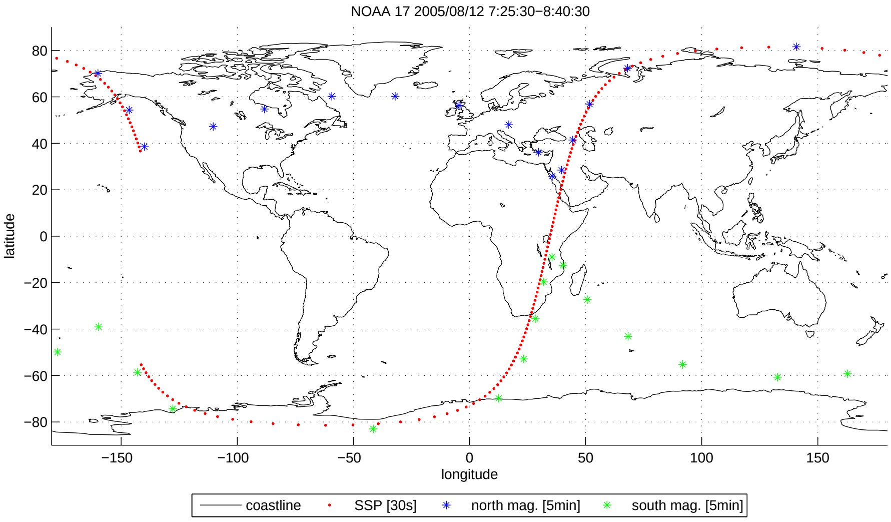


Figure 4.7: Magnetic Conjunction Footprint

The calculated values can be used to study space weather effects. Here in special it will be used in further studies about the cusp regions and to interpolate data between satellite measurements and ground station data.

These functions form the mathematical framework of the toolbox and can be used in various ways to study satellite ground conjunctions. **Vis Sat** combines these three functions in a GUI to provide a first introduction how the functions could be used and how the data could be presented. The GUI and its handling is described briefly in the next section. It should also be mentioned here, that the stand alone functions are downwards compatible with MATLAB® 5.3. The GUI is only written for MATLAB® 7 (Release 14) and is due to changes in the `callback` structure not upwards compatible.

4.2 Graphical User Interface

All function contained in **Vis Sat** are stand alone functions and could be used due to the open-source character and under the GNU General Public License conditions for all satellite applications. In order to get familiar with the functionality and the data calculated by the toolbox functions, **Vis Sat** contains a GUI which enables the user to get a first insight in the handling of the functions and how different input parameter influence the orbit of a satellite. This section describes briefly the GUI and the functionality.

The GUI enables the user to control the complex functions contained in the toolbox by using familiar objects like *"icons"*, *"buttons"* etc. to control the simulation without knowing the exact detailed syntax of every function. It is not necessary to have a deeper insight in the mathematical framework or the numerical model structures to use the GUI to produce first simulation results and to save these results for further use in MATLAB®. The GUI itself is divided into two separate files, the `fig` file which contains the GUI layout (here `vissat.fig`) and the counterpart in an `m` file, this design is used from MATLAB® Version 5.3 Release 12 upwards. The `vissat.m` file implements the function calls for the GUI elements, this function is the interface like the name said between the toolbox functions and the *"icons"*, *"buttons"* and *"radiobuttons"*. The main program `vissat` contains the files `vissat.m` and `vissat.fig` which were created with `GUIDE` (GUI Development Environment) for MATLAB®. This tool can be used in MATLAB® with the call

```
>> guide
```

The tool enables the user to edit directly graphical operation elements and their attributes[8]. The functions which are called by the graphical elements are callback functions, these functions are contained in the counterpart of the `fig` file. The GUI then can be started in MATLAB® with

```
>> vissat
```

MATLAB® combines then automatically the `fig` file layout and the callback functions in the `m` file. All files described here and mentioned above are contained on the attached CD-Rom, here also an HTML documentation of the simulation (Microsoft IE, Netscape and MATLAB® compatible) is attached.

After the call `>> vissat` the GUI shown in fig. 4.8 will appear in a new MATLAB® figure.

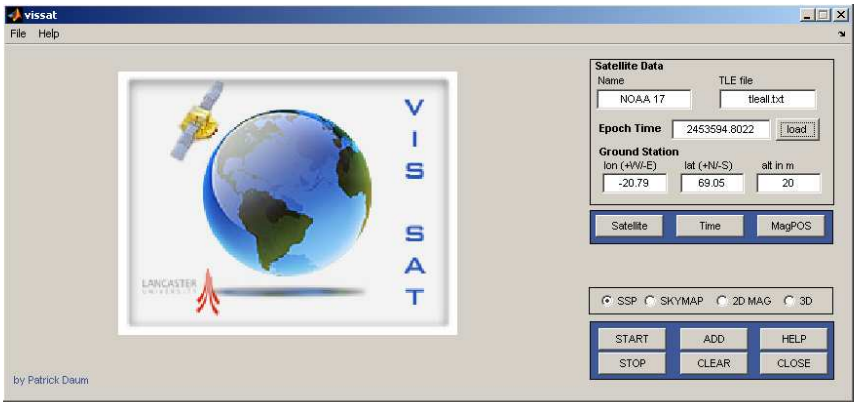

Satellite Data
Name: NOAA 17
TLE file: tleall.txt
Epoch Time: 2453594.8022
Ground Station
lon (+W/-E): -20.79
lat (+N/-S): 69.05
alt in m: 20
● SSP ○ SKYMAP ○ 2D MAG ○ 3D

Figure 4.8: Graphical User Interface startup figure

The control elements on the right side enables the user to set the different parameter which are needed as input of the functions described above. All control elements in

---

[8] for detailed descriptions about the GUI elements see P. Marchand [30] Chapter 10 Elements of GUI Design

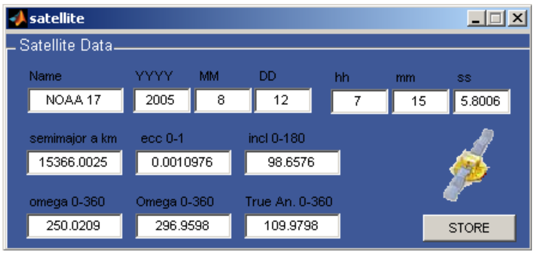
this GUI have an executable part defined in the `m` file. The executable functions in the `m` file have the same MATLAB® syntax as if the stand alone functions would be performed without the framework of the GUI. So all the function calls in the `m` file represent the interface structure of the GUI.

4.2.1 Operation and functionality

The GUI is subdivided in four major parts to control the footprint functions. The first part in the upper right hand corner provides the satellite and ground station data. The first line controls the `norad2kep` function and stores the extracted TLE data in global variables into the workspace of MATLAB® , so it is possible to access these data in all subfunctions of the GUI. To control the function `norad2kep` the name of the satellite and the tle file have to be given. The default values are already stored in the workspace, with the load button the new TLE data will be loaded from the given tle file and stored in the workspace. The epoch time is not settable, when a new TLE data set is loaded it will show the Julian date of this data set. This provides the user with an additional security system to see if the right data are loaded. The ground station position is needed for the skymap plots as input for the `SatTrack` function.

The next part of the GUI enables the user to set satellite data by hand, the simulation time and the parameter for the Tsyganenko model. Each button will open an additional GUI to read in the data. In the following descriptions when it is referred to the main GUI it is the initial `vissat` GUI shown in fig. 4.8.

4.2.1.1 Satellite Input Data



Name: NOAA 17
YYYY: 2005
MM: 8
DD: 12
hh: 7
mm: 15
ss: 5.8006
semimajor a km: 15366.0025
ecc 0-1: 0.0010976
incl 0-180: 98.6576
omega 0-360: 250.0209
Omega 0-360: 296.9598
True An. 0-360: 109.9798

Figure 4.9: Satellite Data

The satellite data interface, shown in fig. 4.9 enables the user to set the main Keplerian elements by hand. The default values are given by the satellite which is specified in the main GUI. In order to avoid data deadlocks, the satellite data GUI will update the given orbital elements only when the name of the satellite is changed. If the name of the satellite is not changed all values are set to the default values again. In order to be compatible with the NORAD format the osculating values given in this GUI have to be transformed to their mean orbital elements. The transformation is based on the processes described in the Spacetrack Report No.3. The behaviour of a typical orbital elements consists of short period, medium period, long period and secular components, this was described in section 2.4.2. Short, medium and long period perturbations are referred to periodics and each refers to the frequency of the respective perturbation. The short periodics have frequencies that are multiples of the satellite's orbital period and long periodics have frequencies which are proportional to the period of the satellite's argument of perigee. The frequency of medium periodics are proportional to the Earth's rotation period relative to a satellite's line of nodes. So short period variations of the orbital elements are functions of the varying mean anomaly through trigonometric relations (see fig. 4.10).

The long period variations are functions of the varying argument of perigee through trigonometric relations. This is explained in detail in the SGP4 and SDP4 model. Here in order to transform the osculating elements into the mean elements which are needed for the model calculations, the classical osculating orbit elements could be described as functions of the mean orbital elements and the short period contributions

$$
\begin{aligned}
a &= a_m + a_{sp}\,(a_m,\, e_m,\, i_m,\, \omega_m,\, M_m) && (4.12)\\
e &= e_m + e_{sp}\,(a_m,\, e_m,\, i_m,\, \omega_m,\, M_m) && (4.13)\\
i &= i_m + e_{sp}\,(a_m,\, e_m,\, i_m,\, \omega_m,\, M_m) && (4.14)\\
\omega &= \omega_m + \omega_{sp}\,(a_m,\, e_m,\, i_m,\, \omega_m,\, M_m) && (4.15)\\
\Omega &= \Omega_m + \Omega_{sp}\,(a_m,\, e_m,\, i_m,\, \Omega_m,\, M_m) && (4.16)\\
M &= M_m + M_{sp}\,(a_m,\, e_m,\, i_m,\, \Omega_m,\, M_m) && (4.17)
\end{aligned}
$$

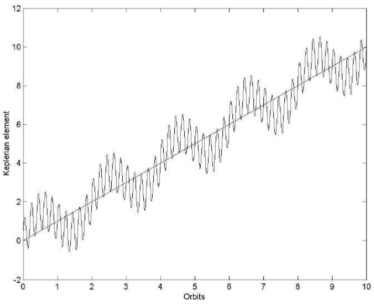


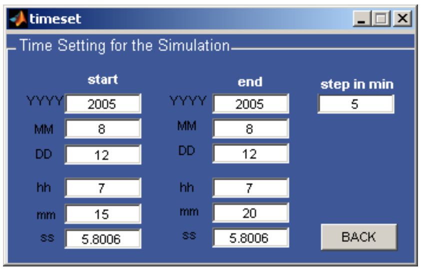
Figure 4.10: Osculating Element Perturbation

The osculation orbital elements define the ideal Keplerian orbit in absence of all outer perturbations. In order to solve this equations for the mean elements in terms of the osculating elements a numerical iteration has to be done. This is done in an subfunction of `satellite.m` and is mainly done by the inversion of the numeric described in the SGP4 and SDP4 model. Due to this iterations it is possible to determine the mean values of the orbit. These values are stored in the `satellite` structure given by the `norad2kep` function. It should be mentioned here, that this method could not calculate the first time derivation of the mean motion, the second time derivation of mean motion and the ballistic coefficient, these SGP4/-SDP4 orbital elements are set to 0 for the calculations, so `sat.tle(2,3,4)=0`. The derivations could be determined with performing a differential correction or orbit determination process but in respect to the computation time this is not implemented.

This GUI allows the user to study footprints of given satellites in an Keplerian notation and to study which effect which parameter has on the orbit. If a satellite orbit should be studied by known mean orbit elements, these have to be stored in a `txt` file in the NORAD standard format and could be read by the main GUI.

4.2.1.2 Time Settings

The functions `SatPOS`, `SatTRACK` and `MagPOS` calculates the position of the footprints for one single time. In order to study the motion of a satellite over a period, the GUI `timeset` enables the user to specify the timeperiod and the stepsize for this period. The default values are the epoch time of the satellite and a range of five minutes with a timestep of 5 minutes. The GUI stores the start and end time of the simulation period in a MATLAB® datevec structure. These times are then passes through the functions to perform the calculations at the given time. In respect to the computation time, the maximum time rage is set to 20 days.

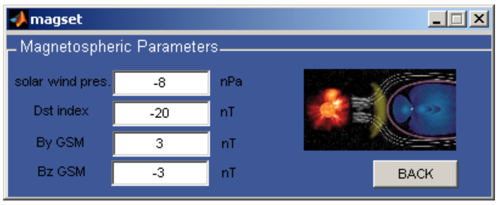


	start		end	step in min
YYYY	2005	YYYY	2005	5
MM	8	MM	8	
DD	12	DD	12	
hh	7	hh	7	
mm	15	mm	20	
ss	5.8006	ss	5.8006	

Figure 4.11: Time Setting

4.2.1.3 Magnetospheric Parameters

The MagPOS button opens the the GUI to set the magnetospheric conditions for the Tsyganenko model. The solar wind pressure in nPa, the DST index in nT, the B_y and B_z-fieldstrength in GSM coordinates in nT. These values describe the magnetic field at the given time range, the default values are set in accordance with [57] and represents mean values.



solar wind pres.	-8	nPa
Dst index	-20	nT
By GSM	3	nT
Bz GSM	-3	nT

Figure 4.12: Magnetospheric Setting

4.2.1.4 Simulation and View

The third and fourth part of the main GUI is to control the simulation and controls the different views. The radiobutton structure corresponds to the main toolbox functions, with `SSP` a two dimensional footprint on a geographical map is produced to show the location of the SSP at the given time. The calculations are done with `SatPOS` and the input parameter for the functions are controlled by the GUI elements mentioned above, the plots produced with these settings could be seen in section 4.1.3. The `SKYMAP` setting produce the visibility plots shown in section 4.1.4. The calculations are done by the function `SatTRACK`. The `2D MAG` setting produce the magnetic footprint in the northern and southern hemisphere on a two dimensional geographical map, the data are calculated by `MagPOS`. The setting `3D` uses the position vector $\vec{r}$ which is returned by the function `SatPOS` to show the satellite and the Earth in a three dimensional view.

The simulation can be controlled by the fourth part the control buttons, with `START` a simulation is started, the call function in the main `vissat.m` file calls the setting parts of the GUI to get the values required for the calculations depending on which radiobutton is set. The `STOP` button interrupts the simulation, this could be done because every process in the GUI is in parallel and not sequential. The `ADD` button allows the user to add another satellite path to the simulated one, this function is only possible if the view setting for the additional satellite is the same as the already presented one. The `CLEAR` button clears the axis element, in case to many plots are made. The `HELP` button starts the online documentation which contains a brief overview of the models implemented in **Vis Sat** and how they could be used.

The description made in this section should be representative for the functionality of the GUI and should be a brief introduction, they described how the interface connects the toolbox functions with graphical control elements. The GUI represents the link between the mathematical complex models and their simple use in the MATLAB® environment by using common control elements. All files for the GUI are attached on the CD-Rom.

4.3 Plot Functions

The plots and the representation of the data in the GUI are produced in MATLAB® 7 (Release 14) with the `map` data included in the main MATLAB® environment. In order to produce the plots also without the GUI, **Vis Sat** contains as support three simple `plot` functions to present the calculated data in a way shown in the sections above and in the GUI. These functions shows how to process and present the data and should support the user to get familiar with the toolbox and with the data format.

4.3.1 Plot on 2D Map (`plot2dmap`)

The `plot2dmap` function represents the data calculated by `SatPOS` and `MagPOS` on a two dimensional geographical map of the Earth. The syntax is

```
>> plot2dmap(lon,lat,border,opt)
```

where `lat` is a vector of latitudes calculated by one of the main functions and `lon` is a vector of calculated longitudes. The third value specifies if political border lines should be shown or not, if border is a positive value the political borders are shown otherwise not. The last value `opt` is a line specification that determines line type, marker symbol, and color of the plotted latitudes and longitudes, default is `'r.'`. The plots shown in section 4.1.3 are produced by this function. In the MATLAB® environment then it is possible to add the plot different additional information analogous to the MATLAB® built in function `plot` (e.g. title, legend etc.).

4.3.2 Plot on Blue Marble Maps (`plotonmarble`)

In order to present the data on high resolution maps like the maps provided by NASA in the Blue Marble project (see [18]) **Vis Sat** contains a function `plotonmarble`.

The syntax is

```
>> plotonmarble(lon,lat,map)
```

where where `lat` is a vector of latitudes calculated by one of the main functions and `lon` is a vector of calculated longitudes. The variable `map` is the bluemarble map as a `jpg`. The function then takes the resolution of the `map` to resize the `lat` and `lon` to fit them onto the map. The default map of this function is `defmarble` which is a map in 2048 × 1024 pixel resolution. Then analogous to the MATLAB® `plot` function different additional information could be added. Figure 4.13 shows the calculated SSP latitudes and longitudes plotted on the default Marble map.

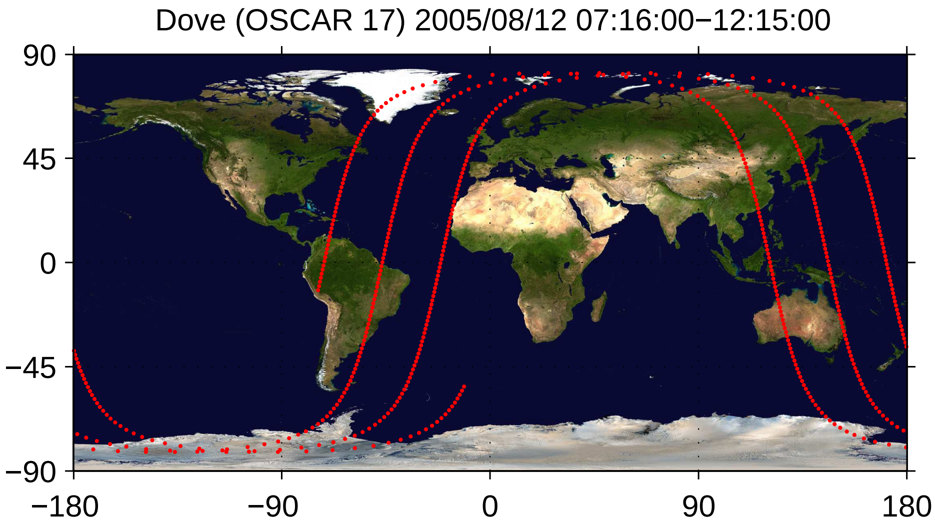


Figure 4.13: Geodetic Position Footprint on Marble

4.3.3 Plot Skymap (`skymap`)

In order to produce skymap plots shown in section 4.1.4 the MATLAB® built in function `polar` can be used, but this function allows not to configure the plotrange, the zero position and various other options which are needed to produce skyplots with the characteristics mentioned in section 4.1.4 and shown in fig. 4.4(a). In order to provide a function to produce skymaps with the charactersitics mentioned above, **Vis Sat** contains a function `skymap`. The syntax is

```
>> skymap(azi,elev)
```

where `azi` is the azimuth angle in degrees and `elev` is the elevation angle in degrees. Negative elevation angles which corresponds to a position of the satellite under the local horizon are not plotted.

4.4 Summary

Vis Sat is designed as a scientific/engineering toolbox in MATLAB® , it provides the functionality of fast calculations of the satellite ground conjunctions. The motion of a satellite is described in principle by the basic celestial laws of motion stated by Kepler and Newton. This mathematical framework which based on the consideration of the forces which acts on the satellite are implemented in a modern simulation environment. The functions provide a tool to calculate satellite footprints and to calculate the visibility. This is accomplished in the functions `SatPOS` and `SatTRACK`, these functions predict the motion of the satellite over the ground and the visibility of the satellite. The functions incorporate SGP4/SDP4 models specified by NORAD in an C/C++ `MEX`-function. This design provides a minimum of computation time and a maximum of flexibility. The functions uses the NORAD TLE data sets to determine the satellite position and velocity at a given time. They provide necessary coordinate transformations and time calculations. The design of the functions and the user interface of each function is kept to a minimum of complexity to provide a manageable tool without knowing the exact mathematical framework. The output of the functions is standardised to MATLAB® and can therefore be used in any kind of application. The main field of application of these functions is in engineering devices controlled by MATLAB® or onboard controller for positioning systems or ground station alignments.

In addition to the simulation of the mechanics the toolbox provides a function to calculate magnetic satellite conjunctions by implementing the Tsyganenko model of the Earth's magnetic field and the routines to trace fieldlines. This model incorporates the IGRF coefficients and give an insight of the magnetic conjunction of the satellite and ground. This functionality is implemented in the function `MagPOS` and incorporates the pre-existing Tsyganenko model.

The main design feature of the toolbox is, that it contains just three main functions. This design removes the complexity of the mathematical models and make them easier to handle. The main focus here was to remove the complexity but without removing functionality at the same time. That is done by the encapsulation of the mathematical models in MATLAB® functions which acts as an interface between the models and the user input, it keeps the user input to a minimum and provides all necessary transformations and additional calculations which are needed for the models. Also these interfaces produce standarised output which could be processed in the MATLAB® environment.

The main function of the toolbox is **Vis Sat** which is the MATLAB® GUI environment to visualise the data produced by the functions described above. It could be used as a scientific analysis tool or as a teaching tool to get a deeper insight in satellite dynamic processes. The visualisation produces two and three dimensional plots of the satellite position. The GUI is started with

```
>> vissat
```

The documentation of the program can be opened in the program itself by the help menu or by

```
>> open vissat.html
```

from the MATLAB® prompt line. The GUI provides a comfortable control for the functions to visualise the calculated data. The function input can be set by the interface and enabled an easy access to the functionality. The GUI produced then two or three dimensional plots of the trajectory of the satellite and two dimensional magnetic conjunction footprints.

5 Conclusion and Future Work

The so called *"space age"* started with the first artificial satellite put into orbit around the planet, nowadays satellites serve for various tasks from communication to broadcasting, imaging or scientific purposes. All these satellites obey the rules of celestial mechanics, to visualise the motion in respect to these rules and to develop a software tool for analysis of the satellite ground conjunctions was the main aim of this thesis.

The thesis provides the theoretical background which is successfully implemented, tested and validated in the toolbox **VIS SAT**. It can be used to visualise the orbit trajectory of a satellite under different points of view. It allows the user to study the satellite ground conjunctions by tracking the satellite over the ground or by calculating the elevation and azimuth angle of the satellite from a static ground station position. It further enables the user to study magnetic conjunctions by calculating magnetic footprints. These functionalities can be used on the one hand from an engineering point of view to adjust ground station dishes to establish reliable links to the satellite by calculating the exact position by using the TLE data sets. This was also a reason why MATLAB® as an implementation environment was used, the functions in **VIS SAT** can be reused for different kind of tasks on a common basis. On the other hand **VIS SAT** can be used to simulate an orbit of a satellite in advance to see which orbit parameters the satellite has to have in order to fulfill its task. Here the visualisation on the one hand supports the numerical functions in order to get a first impression of the motion and it also helps to clarify the physical processes which are the bases of the motion. The magnetic conjunction simulation can be used for further space weather studies, here in special it will be used to correlate Cluster satellite data with ground station data in order to verify the magnetic conditions in a vertical cut through the cusp regions.

For the satellite dynamics the SGP4 and SDP4 models specified by NORAD were implemented, for the magnetic conjunctions the Tsyganenko model (Version 2003) was implemented. These models have several advantages and disadvantages. The advantages of the SGP4 and SDP4 models, which based on the fundamental laws stated by Newton and Kepler, are that these models have been verified by NORAD and provide a precise and manageable mathematical framework for the orbit calculations. Also here the implementation in C/C++ by wrapping the C/C++ code in an `MEX` function provides a maximum of flexibility and ensures that future specifications by NORAD can be implemented easily. However these models work with mean values, NORAD has removed periodic variations in a particular way, and the models in their present form does not contain numerical integration methods. The magnetic conjunction simulation is done by implementing the Tsyganenko model (Version 2003) of the Earth's magnetic field. This model provides a numerical description of the magnetosphere and incorporates more than 28,000 measurements. This framework presents the state of the art and is implemented straight forward in MATLAB®. Here some more details are mentioned in the future work part. However, this model needs to be updated every five years due to the fact that it is based on th IGRF coefficients. Nevertheless with these implementation in **VIS SAT** it is possible to calculate the magnetic conjunction, namely the latitude and longitude of the footprints in the northern and southern hemisphere respectively.

All functions of **Vis Sat** described in this thesis are executable in a command line way and can therefore be used in various application, these is mainly the field for experienced users to use **Vis Sat** as a scientific analysis tool. The GUI of **Vis Sat** presents an environment which also can be used by less experienced users in order to get a deeper insight of the physics behind the motion described in the theoretical part of this theses. The GUI provides easy access to the complex software systems. All functions are based on numerical calculations and the visualisation of the output of these functions to represent the real world in a graphical controllable environment [30]. The open-source structure of **Vis Sat** allows users to incorporate the toolbox or parts of it in any kind of application in satellite dynamics and space environment studies.

Future Work

The main field of application for this thesis is to calculate and visualise satellite/ground conjunctions and to visualise the motion of the satellite. The tool **Vis Sat** can be used to do this and to use the functions for further studies of the satellite ground conjunctions. This studies are just limited by the functionality of the functions and the outer boundaries of the incorporated models. In order to obtain more precise orbit and conjunction calculation further work should be done in the following field

- accurate geopotential; the geopotential model is changing due to new satellite measurements and due to new numerical models. So here an update of the function and especially the zonal harmonics have to be done.

- aerodynamic forces; to predict the aerodynamic force is very difficult and it depends on a lot of parameters, so the SGP model just introduced a mean drag force term, here some new basic approaches could be implemented (see details [33])

- slow implementation; the `GEOPACK` routines are very slow. In **Vis Sat** a direct implementation of the `GEOPACK` is used. For future work it has to be mentioned, that these implementation is slow due to the loop structures in it and it would be recommendable to write an `MEX` function to wrap the FORTRAN code. Here it was not done, because of the exception handler in MATLAB® 7. The MATLAB® 7 version uses C/C++ exceptions in library functions, this means that the `MEX` function from FORTRAN to MATLAB® has to propagate exceptions in a C/C++ manner. So in order to get a faster simulation this problem has to be solved, either in solving the incompatibility between the MATLAB® FORTRAN gateway or by implementing the `GEOPACK` in C/C++ .

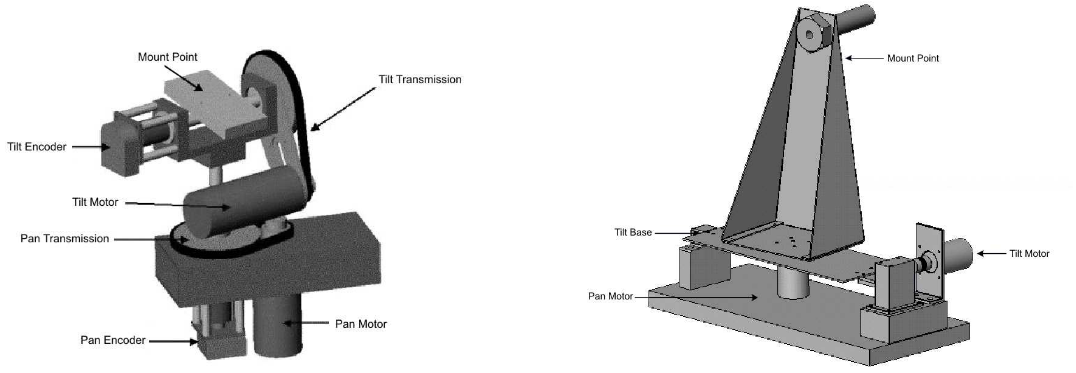
- orbit precision; the embedded orbit dynamic models do not incorporate numerical integration steps for high precision orbit propagation (like the GODDAT SSC Locator), as a future task this could be implemented following the descriptions in [33] (new edited Version published August 2005) also by implementing this high precision integrators the time period in which the models are correct could be enlarged. Here at the current implementation state it is recommendable to limit the simulation time to three month in order to take the maximum precision.

VIS SAT is a general visualisation tool, the calculations are based on the TLE data sets, in order to increase the precision of the propagation, sophisticated sets of high-fidelity force models and numerical integration techniques should be implemented but these models are based on finite element modeling structures which would increase the complexity of the system and would make it unmanageable to handle in a MATLAB® environment. The problem described with the FORTRAN `MEX` function results in a slow straight forward MATLAB® implementation which makes it necessary to abandon an option to visualise the magnetic field lines in a three dimensional view in respect to the computation time.

These further investigations and necessary updates will speed up the propagation and will make it more precise but it should be mentioned with every step which makes the propagation more precise it will increase the complexity. So for future work it is recommendable to have in mind the balance between complexity and how precise it must be, also with every step in complexity the computation time will increase.

Applications

VIS SAT in its current implementation is an open-source scientific toolbox in MATLAB® with the intended purpose to give an insight in satellite ground conjunctions. The data produced with the functions could be used for different purposes, here a particular application should be mentioned. The flight dynamics calculations could be used to control a tracking device operated by MATLAB® . Such a device was build in the Department of Engineering at Lancaster University during a project undertaken from O.R. Ariyo (Adaptive Sun-Follower for Building Illumination, Lancaster University, Department of Engineering, 2005). The tracking device is meant to be a Sun tracker but the main engineering tracker device is controlled by calculated elevation and azimuth angles, with the function `SatTRACK` these values could be calculated for all space objects included in the TLE data. So it is possible to use the main tracking device with a telescope or antenna operated by MATLAB® with the calculated values of `SatTRACK`. So it would be possible to track the path over the horizon for imagery, observation or communication purposes nearly in real time. This is just one example of an real hardware application but it is representative for how the numerical calculations in the MATLAB® environment could be used to control a hardware device (example device see fig. 5.1).



(a) tracking device with belt transmission (b) tracking device with direct transmission

Figure 5.1: Tracking Device

References

[1] Baker, D.N., *"How to cope with space weather"*, Science, **297**, 2002.

[2] Betts, J.T., *"Optimal Interplanetary Orbit Transfers by Direct Transcription"*, The Journal of the Astronautical Sciences, **Vol. 42**, No. 3, pp. 247-268, 1994.

[3] Binebrink, A.L., Radomski, M.S., Samii, M.V., *"Atmospheric drag model calibrations for spacecraft lifetime prediction"*, In NASA, Goddard Space Flight Center, Flight Mechanics/Estimation Theory Symposium, pp. 445-458, 1989.

[4] Born, G.H., *"Motion of a Satellite under the influence of an oblate Earth (lecture Notes ASEN 3200)"*, Colorado Center for Astrodynamics Research, 2001.

[5] Brouwer, D., *"Solution of the problem of artifcial satellite theory without drag"*, Astronomical Journal, **64**, pp. 378-398, 1959.

[6] Bunnell, P., *"Tracking Satellites in Elliptical Orbits"*, Ham Radio magazine, pp. 46-50, March 1981.

[7] Carvahlo, G.B., *"Reference Frame Definitions"*, Technical Report, INT-GEN-DF-ZAR-001, ZARM Bremen, 2003.

[8] CSEM Center for Space Environment Modeling - 3D modeling of the Earth's magnetic field under different solar wind conditions 2004 (available under: http://csem.engin.umich.edu stand: September 7, 2005)

[9] Danby, J.M., *"Fundamentals of Celestial Mechanics, 2nd ed."*, VA: Willmann-Bell, 1988.

[10] Daum, P., Heber, B., *"Erdmagnetfeld, Grundlagen und assoziierte Phänomene"*, Seminar Paper, Universität Osnabrück, 2004.

[11] Daum, P., *"Visualisierung der Bewegung geladener Teilchen in elektromagnetischen Feldern"*, Bachelor Thesis, Universität Osnabrück, 2004.

[12] Davidoff, M.R., *"Satellite Experimenter's Handbook, 2nd Edition"*, American Radio Relay League, 1990.

[13] Davies, J., K., *"Satellite Astronomy, the principles and practice of astronomy from space"*, Ellis Horwood Limited, 1988.

[14] Fitzpatrick, P.M., *"Principles of Celestial Mechanics"*, Academic Press, 1970.

[15] Glatzmaier, G., *"3D Numerical Simulation of the Geodynamo and the Earth's Magnetic Field"*, Pittsburgh Supercomputing Center, 2000.

[16] Grafarend, E.W., *"Space-time geodesy"*, Bolletino Di Geodesia e Scienze Affini, **XXXVIII No.2**, pp. 305-343, 1979.

[17] Grafarend, E.W., *"Introductory lectures to geodesy, vol. 1"*, Universität Stuttgart Geodätisches Institut (GIS), 1999.

[18] Herring, D., *"The Blue Marble, NASA earth observatory"*, distributed by NASA, (available under: http://earthobservatory.nasa.gov/Newsroom/BlueMarble/ stand: September 7, 2005).

[19] Jomann, U., *"Time Equivalence of the Tropical Year and the Sidereal Year"*, Journal of Theoretics, **Vol. 3-3**, 2001.

[20] Hoots, F.R., Roehrich, R.L., *"Spacetrack Rep. No.3: Models for Propagation of NORAD Element Set"*, Compiled and distributed by TS Kelso, December 1980 (available under: http://www.celestrak.com/NORAD/documentation/spacetrk.pdf stand: September 7, 2005).

[21] Hujsak, R.S., *"A Restricted Four Body Solution for Resonating Satellites with an Oblate Earth"*, AIAA Paper No. 79-136, 1979.

[22] Hujsak, R.S., Hoots, F.R., *"Deep Space Perturbations Ephemeris Generation"*, Aerospace Defense Command Space Computational Center Program Documentation, **DCD 8**, Section 3, 82-104, 1977.

[23] International Association of Geomagnetism and Aeronomy. International geomagnetic reference field - epoch 2005, 2005 (available under: http://www.ngdc.noaa.gov/IAGA/wg8/igrf2000.html stand: September 7, 2005).

[24] Kaula, W.M., *"Theory of Satellite Geodesy"*, Dover Publicatians Inc., 2000.

[25] Kelso, T.S., *"Orbital Coordinate Systems, Part I"*, Satellite Times, 2, **No. 1**, pp. 80-81, September/October 1995.

[26] Kelso, T.S., *"Orbital Coordinate Systems, Part II"*, Satellite Times, 2, **No. 2**, pp. 78-79, November/December 1995.

[27] Kelso, T.S., *"Orbital Coordinate Systems, Part III"*, Satellite Times, 2, **No. 3**, pp. 78-79, January/February 1996.

[28] King-Hele, D.G., *"Observing Earth Satellites"*, Macmillan , 1983.

[29] Lyddan, R.H., *"Small Eccentricities or Inclinations in the Brouwer Theory of the Artificial Satellite"*, Astronomical Journal, **68**, pp. 555-558, 1963.

[30] Marchand, P., Holland, O.T., *"Graphics and GUIs with MATLAB® Ed. III"*, Chapman & Hall/CRC, 2003.

[31] Maday, M., *"Grundlagen und Software für die Bahnberechnung von Satelliten"*, beam-Verlag, 1995.

[32] Miles, H., *"Artificial Satellite Observing and its Applications"*, Faber and Faber Limited, 1974.

[33] Montenbruck, O., Gill, E., *"Satellite Orbits"*, Springer, 2003.

[34] Moulton, F.R., *"An Introduction to Celestial Mechanics"*, Dover Publicatians Inc., 1970.

[35] Paul, E.N., Sylvio, F.-M., *"Natural and Artificial Satellite Motion"*, University of Texas Press, 1979.

[36] Pfister, C., Grafarend, E.W., Schäfer, C., *"Gravitative Einflsse anderer Himmelskörper auf Satellitenbahnen mit Umsetzung in eine MATLAB-Funktion"*, Universität Stuttgart Geodätisches Institut (GIS), 1999.

[37] Pritchard, W.L., *"Satellite Communication Systems Engineering"*, Prentice Hall Inc., 1993.

[38] Prölss, G.W., *"Physik des erdnahen Weltraums"*, Springer, 2001.

[39] Pulkkinen, T.I., Tsyganenko, N.A., *"Testing the accuracy of magnetospheric model field line mapping"*, J. Geophys. Res., **Vol. 101(A12)**, pp. 27431-27442, 1996.

[40] Quarteroni, A., Saleri, F., *"Scientific Computing with MATLAB"*, Springer, 2003.

[41] Roy, A.E., *"Orbital Motion"*, Adam Hilger Ltd., 1978.

[42] Russell, C.T., *"Geophysical Coordinate Transformations"*, Cosmic Eletrodynamics, **Vol. 2**, pp. 184-196, 1971.

[43] Schäfer, C., *"Space gravity spectroscopy. The sensitivity analysis of GPS-tracked satellite missions (case study CHAMP) (Doctoral Thesis, Universität Stuttgart, 2000)"*, Verl. der Bayer. Akad. der Wiss., 2001.

[44] Scherer, K., Fichtner, H., Heber, B., Mall, U., *"Space Weather, Lect. Notes Phys. 656"*, Springer, 2005.

[45] Smart, W.M., *"Spherical Astronomy"*, Cambridge University Press, 1962.

[46] Stern, D.P., *"The art of mapping the magnetosphere"*, J.Geophys.Res., **Vol. 99**, pp. 17169-17198, 1994.

[47] Thompson, P.D., *"A General Technique for Satellite Tracking"*, QST - ARRL, pp. 29-34, November 1975.

[48] Tipler, P.A., Mosca, G., Pelte, D., *"Physik"*, Spektrum Akademischer Verlag, 2004.

[49] Tsyganenko, N.A., Usmanov, A.V., *"Determination of the Magnetospheric Current System Parameters and Development of Experimental Geomagnetic Field Models Based on Data from IMP and HEOS Satellites"*, Planet. Space Sci., **Vol. 30**, pp. 985-998, 1982.

[50] Tsyganenko, N.A., Usmanov, A.V., Papitashvili, V.O., Papitashvili, N.E., Popov, V.A., *"Software for Computations of Geomagnetic Field and Related Coordinate Systems"*, Soviet Geophysical Committee, **Special Report**, 58 pp., Moscow, 1987.

[51] Tsyganenko, N.A., *"Global Quantitative Models of the Geomagnetic Field in the Cislunar Magnetosphere for Different Disturbance Levels"*, Planet. Space Sci., **Vol. 35**, pp. 1347-1358, 1987.

[52] Tsyganenko, N.A., *"A Magnetospheric Magnetic Field Model with a Warped Tail Current Sheet"*, Planet. Space Sci., **Vol. 37**, pp. 5-20, 1989.

[53] Tsyganenko, N.A., *"Modeling the Earth's Magnetospheric Magnetic Field Confined Within a Realistic Magnetopause"*, J.Geophys.Res., **Vol. 100**, pp. 5599-5612, 1995.

[54] Tsyganenko, N.A., Stern, D.P., *"Modeling the Global Magnetic Field of the Large-Scale Birkeland Current Systems"*, J. Geophys.Res., **Vol. 101**, pp. 27187-27198, 1996.

[55] Tsyganenko, N.A., *"Magnetic Field Model, FORTRAN Routines, Geopack 2005"*, NASA GSFC, 2005 (available under: http://nssdcftp.gsfc.nasa.gov/models/magnetospheric/tsyganenko/ and http://nssdc.gsfc.nasa.gov/space/model/magnetos/data-based/modeling.html stand: September 7, 2005)

[56] Wertz, J.R., Larson, W.J., "Space mission analysis and design", Kluwer academic publishers, 1999.

[57] Xilian, L., Temerin, M., Baker, D.N., Reeves, G.D., Larson, D., Kanekal, S.G., *"The Predictability of the Magnetosphere and Space Weather"*, AGU, EOS, **Vol. 84**, No. 37, 2003.

A Equations, Tables, Explanations

A.1 Geopotential (gravitational force)

Gravity is the driving force of this simulation. Newton's law of universal gravitation holds for a spherically symmetric mass equation (2.6), but the Earth is known to neither have an even mass distribution nor be a perfect sphere. An accurate model incorporating the variation of the Earth's gravity at different points in space which can be established based on observational data. This data has been obtained from several sources

- satellite ranging lasers that utilise doppler shift to measure satellite positions in the order of cm, which are used determine the gravity experienced by the satellites;
- spring gravimeters, located on the Earth's surface, that measure the local gravitational acceleration with an accuracy of 10^{-8} m/s^2;
- satellite-based altimeters, that can be used to determine the mean sea surface level, which can in turn be used to provide information about the shape of the Earth and thus geopotential coefficients.

To account for the deviation from equation (2.6), the Earth's gravity potential (or geopotential") is written as an expansion in a series of Legendre polynomials.

It can be written for cartesian coordinates of acceleration due to gravity in a ECEF frame

$$\ddot{x} = \sum_{n,m} \ddot{x}_{nm} \qquad \ddot{y} = \sum_{n,m} \ddot{y}_{nm} \qquad \ddot{z} = \sum_{n,m} \ddot{z}_{nm} \tag{A.1}$$

with

$$\begin{aligned}
\ddot{x}_{nm} &\overset{m=0}{=} \frac{GM}{|r|^2}\left(-C_{n0}V_{n+1,1}\right) \\
&\overset{m>0}{=} \frac{GM}{2|r|^2}\left((-C_{nm}V_{n+1,m+1} - S_{nm}W_{n+1,m+1}) + \right. \\
&\qquad\qquad \left. \frac{(n-m+2)!}{(n-m)!}\left(C_{nm}V_{n+1,m-1} + S_{nm}W_{n+1,m-1}\right)\right) && \text{(A.2)} \\
\ddot{y}_{nm} &\overset{m=0}{=} \frac{GM}{|r|^2}\left(-C_{n0}W_{n+1,1}\right) \\
&\overset{m>0}{=} \frac{GM}{2|r|^2}\left((-C_{nm}W_{n+1,m+1} + S_{nm}V_{n+1,m+1}) + \right. \\
&\qquad\qquad \left. \frac{(n-m+2)!}{(n-m)!}\left(-C_{nm}W_{n+1,m-1} + S_{nm}V_{n+1,m-1}\right)\right) && \text{(A.3)} \\
\ddot{z}_{nm} &= \frac{GM}{|r|^2}\left((n-m+1)\left(-C_{nm}V_{n+1,m} - S_{nm}W_{n+1,m}\right)\right) && \text{(A.4)}
\end{aligned}$$

The V_{mn} and W_{nm} terms satisfy the recurrence relation $V_{00} = \vec{r}/|r|$ and $W_{00} = 0$

$$V_{mm} = (2m-1)\left(\frac{x\vec{r}}{|r|^2}V_{m-1,m-1} - \frac{y\vec{r}}{|r|^2}W_{m-1,m-1}\right) \tag{A.5}$$

$$W_{mm} = (2m-1)\left(\frac{x\vec{r}}{|r|^2}W_{m-1,m-1} - \frac{y\vec{r}}{|r|^2}V_{m-1,m-1}\right) \tag{A.6}$$

and

$$V_{nm} = \frac{2n-1}{n-m}\frac{z\vec{r}}{|r|^2}V_{n-1,m} - \frac{n+m-1}{n-m}V_{n-2,m} \tag{A.7}$$

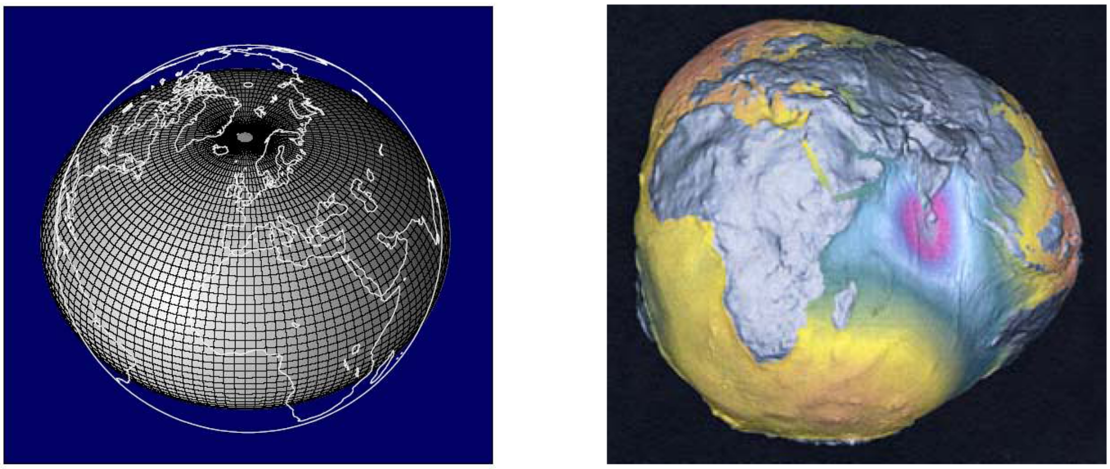
$$W_{nm} = \frac{2n-1}{n-m}\frac{z\vec{r}}{|r|^2}W_{n-1,m} - \frac{n+m-1}{n-m}W_{n-2,m}\,. \tag{A.8}$$

The C_{nm} and S_{nm} coefficients are tabulated for different models. The implementation of the stable numerical model could be seen in [33]. Figure A.1 shows the different model distribution for this geopotential.

Depending upon its degree n and order m a Legendre function is referred to

- a zonal harmonic
- a sectoral harmonic
- a tesseral harmonic

A zonal harmonic corresponds geometrically to a particular shape of the geopotential surface. The second zonal harmonic which expresses the main effect of the Earth's flattening J_2, makes a north-south slice through the Earth appear elliptical; the third zonal harmonic J_3 provides a profile with a tendency to a triangle.; the fourth J_4 harmonic relating to a square (details see [4]).



(a) static oblateness of the Earth, model from 1957

(b) complex model of the geopotential 2002

Figure A.1: Geopotential (source: GFZ Potzdam)

A.2 Earth Atmosphere (drag force)

Knowledge of the atmospheric density at the satellite's position is required to calculate the drag force on the satellite. It may also be required in terms of establishing a reliable link to the satellite, however this may require more information than density alone [28].

Accurate modeling of aerodynamic forces is difficult for the following reasons

- the density of the upper atmosphere is not known very accurately;
- modeling the drag force requires detailed knowledge of the interaction of neutral gas, as well as charged particles, with the different spacecraft surfaces;
- the changing attitude of a satellite will present different geometries and surfaces to the oncoming flow.

Just for density calculations two models should be mentioned here the Harris-Priester model and the Jacchia model, for more details see [3].

A.3 Time-Variant Kepler Elements

The orbit of a satellite in the central force field of the Earth could be described by the six Keplerian elements, when disturbance force are also considered these elements become time-variant (cf. [2, 24]). It can be written

$$\frac{\partial a}{\partial t} = \frac{2e\,\sin(\nu)}{nx}F_r + \frac{2ax}{nr}F_s \tag{A.9}$$

$$\frac{\partial e}{\partial t} = \frac{x\,\sin(\nu)}{na} + \frac{x}{na^2e}\left(\frac{a^2x^2}{r} - r\right)F_s \tag{A.10}$$

$$\frac{\partial i}{\partial t} = \frac{r\,\cos(u)}{na^2x}F_w \tag{A.11}$$

$$\frac{\partial \Omega}{\partial t} = \frac{r\,\sin(u)}{na^2x\,\sin(i)}F_w \tag{A.12}$$

$$\frac{\partial \omega}{\partial t} = -\frac{x\,\cos(\nu)}{nae}F_r + \frac{p}{ev}\left[\sin(\nu)\left(1 + \frac{1}{1+e\,\cos(\nu)}\right)\right]F_s - -\frac{r\,\cot(i)\sin(u)}{na^2x}F_w \tag{A.13}$$

$$\frac{\partial \nu}{\partial t} = \frac{x\,\cos(\nu)}{nae}F_r - \frac{p}{eh}\sin(\nu)\left(1+\frac{r}{p}\right)F_s + +\sqrt{\frac{GM}{p^3}}\left(1+e\,\cos(\nu)\right)^2 \tag{A.14}$$

using the following definitions

$$x = \sqrt{1-e^2} \qquad p = a\left(1-e^2\right) \qquad v = \sqrt{GMp}$$

$$n = \sqrt{\frac{GM}{a^3}} \qquad r = \frac{p}{1-e\,\cos(\nu)} \qquad u = \nu + \omega$$

It can be seen from Equations A.9-A.14 that the derivatives of the first five elements become zero, if no disturbance torques are present. The satellite is on a Keplerian orbit again. Only the true anomaly ν has a part that is independent of disturbance torques and represents the normal orbital motion. Since the Equations have several singularities for circular $e = 0$ and for equatorial $i = 0°$ orbits. For an actual integration the equinoctial elements should be used as described in [2] and [31]. They can be calculated from the classical element set but do not have the disadvantage of the singularities.

A.4 Detailed SGP Model

This section deals with the original SGP model described in the Spacetrack Report No. 3 [20] by F.R. Hoots and R.L. Roehrich. Here the connection between the SGP components and the Keplerian elements in section 2.3.4 is made. The SGP model takes the original Kepler elements and modifies these elements in accordance to the additional forces which acts on the satellites. With the modification of the Keplerian elements it is possible to correct the Kepler prediction and to produce exact propagations.

A.4.1 SGP Model

The NORAD mean element sets can be used for prediction with SGP, these model takes the drag effect on mean motion as linear in time. Predictions are made by first calculating the constants (here only the necessary formulae for the orbit motion description are shown)

$$a_1 = \left(\frac{GM}{n_0}\right)^{2/3} \tag{A.15}$$

$$\delta_1 = \frac{3}{4} J_2 \frac{R_E^2}{a_1^2} \frac{\left(3\cos^2(i_0) - 1\right)}{\left(1-e_0^2\right)^{3/2}} \tag{A.16}$$

$$a_0 = a_1 \left(1 - \frac{1}{3}\delta_1 - \delta_1^2 - \frac{134}{81}\delta_1^3\right) \tag{A.17}$$

$$p_0 = a_0\left(1 - e_0^2\right) \tag{A.18}$$

$$\frac{\partial \Omega}{\partial t} = -\frac{3}{2} J_2 \frac{R_E^2}{p_0^2} n_0 \cos(i_0) \tag{A.19}$$

$$\frac{\partial \omega}{\partial t} = \frac{3}{4} J_2 \frac{R_E^2}{p_0^2} n_0 \left(5\cos^2(i_0) - 1\right) \tag{A.20}$$

The secular effects of atmospheric drag and gravitation are included through the following equations

$$a = a_0 \left(\frac{n_0}{n_0 + 2\left(\dot{n}_0/2\right)(t-t_0) + 3\left(\ddot{n}_0/6\right)(t-t_0)^2}\right)^{2/3} \tag{A.21}$$

$$e = \begin{cases} 1 - \left(a_0\left(1-e_0\right)/a\right. & \text{for } a > a_0\left(1-e_0\right) \\ 10^{-6} & \text{for } a \leq a_0\left(1-e_0\right) \end{cases} \tag{A.22}$$

$$p = a\left(1 - e^2\right) \tag{A.23}$$

$$\Omega_S = \Omega_0 + \frac{\partial \Omega}{\partial t}(t - t_0) \tag{A.24}$$

$$\omega_S = \omega_0 + \frac{\partial \omega}{\partial t}(t - t_0) \tag{A.25}$$

Long-period periodics are includes through the following equations

$$a_{x\,\mathrm{NSL}} = e\cos(\omega_S) \tag{A.26}$$

$$a_{y\,\mathrm{NSL}} = e\sin(\omega_S) - \frac{1}{2}\frac{J_3}{J_2}\frac{R_E}{p}\sin(i_0) \tag{A.27}$$

Then it is necessary to solve the Kepler's equation (2.41) with these new calculated values, where

$$(E+\omega)_{k+1} = (E+\omega)_k + \Delta(E+\omega)_k \tag{A.28}$$

with

$$\begin{aligned}
(E+\omega)_1 &= U && \text{(A.29)}\\
\Delta(E+\omega)_k &= \frac{U - a_{y\,\mathrm{NSL}}\cos(E+\omega)_k + a_{x\,\mathrm{NSL}}\sin(E+\omega)_k - (E+\omega)_k}{-a_{y\,\mathrm{NSL}}\sin(E+\omega)_k - a_{x\,\mathrm{NSL}}\cos(E+\omega)_k + 1} && \text{(A.30)}
\end{aligned}$$

where

$$\begin{aligned}
U &= (M_0+\omega_0+\Omega_0) + \left(n_0 + \frac{\partial\omega}{\partial t} + \frac{\partial\Omega}{\partial t}\right)(t-t_0) + \frac{\dot{n}_0}{2}(t-t_0)^2 + \frac{\ddot{n}_0}{6}(t-t_0)^3 - \\
&\quad -\frac{1}{4}\frac{J_3}{J_2}\frac{R_E}{p}\,a_{x\,\mathrm{NSL}}\,\sin(i_0)\left(\frac{3+5\cos(i_0)}{1+\cos(i_0)}\right) - \Omega_S && \text{(A.31)}
\end{aligned}$$

Then the intermediate (partially osculating) quantities can be calculated by

$$\begin{aligned}
e\cos(E) &= a_{x\,\mathrm{NSL}}\,\cos(E+\omega) + a_{y\,\mathrm{NSL}}\,\sin(E+\omega) && \text{(A.32)}\\
e\sin(E) &= a_{x\,\mathrm{NSL}}\,\sin(E+\omega) - a_{y\,\mathrm{NSL}}\,\cos(E+\omega) && \text{(A.33)}\\
e_L^2 &= a_{x\,\mathrm{NSL}}^2 + a_{y\,\mathrm{NSL}}^2 && \text{(A.34)}\\
p_L &= a\left(1-e_L^2\right) && \text{(A.35)}\\
r &= a\,(1-e\cos(E)) && \text{(A.36)}\\
\dot{r} &= GM\,\frac{\sqrt{a}}{r}\,e\sin(E) && \text{(A.37)}\\
r\dot{\nu} &= GM\,\frac{\sqrt{p_L}}{r} && \text{(A.38)}\\
\sin(u) &= \frac{a}{r}\left(\sin(E+\omega) - a_{y\,\mathrm{NSL}} - a_{x\,\mathrm{NSL}}\,\frac{e\sin(E)}{1+\sqrt{1-e_L^2}}\right) && \text{(A.39)}\\
\cos(u) &= \frac{a}{r}\left(\cos(E+\omega) - a_{x\,\mathrm{NSL}} + a_{y\,\mathrm{NSL}}\,\frac{e\sin(E)}{1+\sqrt{1-e_L^2}}\right) && \text{(A.40)}\\
u &= \tan^{-1}\left(\frac{\sin(u)}{\cos(u)}\right) && \text{(A.41)}
\end{aligned}$$

Short-period perturbations are now included by

$$\begin{aligned}
r_k &= r + \frac{1}{4}\,J_2\,\frac{R_E^2}{p_L}\,\sin^2(i_0)\cos(2u) && \text{(A.42)}\\
u_k &= u + \frac{1}{8}\,J_2\,\frac{R_E^2}{p_L^2}\,\left(7\cos^2(i_0)-1\right)\sin(2u) && \text{(A.43)}\\
\Omega_k &= \Omega_S + \frac{3}{4}\,J_2\,\frac{R_E^2}{p_L^2}\,\cos(i_0)\sin(2u) && \text{(A.44)}\\
i_k &= i_0 + \frac{3}{4}\,J_2\,\frac{R_E^2}{p_L^2}\,\sin(i_0)\cos(i_0)\cos(2u) && \text{(A.45)}
\end{aligned}$$

Then in accordance with the flow chart on page 17 the corrected unit orientation vectors could be calculated by substituting the sin( ) and cos( ) terms of ν by using u_k. Then position and velocity are given by

$$\vec{r} = r_k\vec{P} \qquad \text{and} \qquad \dot{\vec{r}} = \dot{r}\vec{P} + (r\dot{\nu})\,\vec{Q} \qquad \text{(A.46)}$$

A.4.2 SGP4 Model

This model is an extension of the original SGP model and takes into account that the drag effects not linear on the mean motion. Therefore it is necessary to define where the perigee is in order to take the different drag effects into account.

$$a_1 = \left(\frac{GM}{n_0}\right)^{2/3} \tag{A.47}$$

$$\delta_1 = \frac{3}{4} J_2 \frac{R_E^2}{a_1^2} \frac{\left(3\cos^2(i_0) - 1\right)}{\left(1 - e_0^2\right)^{3/2}} \tag{A.48}$$

$$a_0 = a_1 \left(1 - \frac{1}{3}\delta_1 - \delta_1^2 - \frac{134}{81}\delta_1^3\right) \tag{A.49}$$

$$\delta_0 = \frac{3}{4} J_2 \frac{R_E^2}{a_0^2} \frac{\left(3\cos^2(i_0) - 1\right)}{\left(1 - e_0^2\right)^{3/2}} \tag{A.50}$$

$$n_0' = \frac{n_0}{1 + \delta_0} \tag{A.51}$$

$$a_0' = \frac{a_0}{1 - \delta_0} \tag{A.52}$$

$$s^* = \begin{cases} a_0'(1 - e_0) - 78 + R_E & 98 < \text{ perigee } \leq 156 \\ 20/6378.135 + R_E & \text{perigee } \leq 98 \end{cases} \tag{A.53}$$

$$q_0 = a_0 (1 - e_0) \tag{A.54}$$

Then the constants can be calculated by using the appropriate s

$$\Phi = \cos(i_0) \tag{A.55}$$

$$\zeta = \frac{1}{a_0' - s} \tag{A.56}$$

$$\beta_0 = \left(1 - e_0^2\right)^{1/2} \tag{A.57}$$

$$\eta = a_0' e_0 \zeta \tag{A.58}$$

$$\begin{aligned} C_2 = {} & (q_0 - s)^4 \zeta^4 n_0' \left(1 - \eta^2\right)^{-7/2} \left(a_0' \left(1 + \frac{3}{2}\eta^2 + 4e_0\eta + e_0\eta^3\right) + \right. \\ & \left. + \frac{3}{4} J_2 \frac{R_E^2 \zeta}{(1 - \eta^2)} \left(-\frac{1}{2} + \frac{3}{2}\Phi^2\right) \left(8 + 24\eta^2 + 3\eta^4\right)\right) \end{aligned} \tag{A.59}$$

$$C_1 = B^* C_2 \tag{A.60}$$

$$C_3 = -\frac{2 (q_0 - s)^4 \zeta^5 J_3 R_E n_0' \sin(i_0)}{J_2 e_0} \tag{A.61}$$

$$\begin{aligned} C_4 = {} & 2n_0' (q_0 - s)^4 \zeta^4 a_0' \beta_0^2 \left(1 - \eta^2\right)^{-7/2} \left(\left(2\eta(1 + e_0\eta) + \frac{1}{2}e_0 + \frac{1}{2}\eta^3\right) - \frac{J_2 R_E^2}{a_0'(1 - \eta^2)} \cdot \right. \\ & \left(3\left(1 - 3\Phi^2\right)\left(1 + \frac{3}{2}\eta^2 - 2e_0\eta - \frac{1}{2}e_0\eta^3\right) + \frac{3}{4}\left(1 - \Phi^2\right) \right. \\ & \left.\left. \left(2\eta^2 - e_0\eta - e_0\eta^3\right)\cos(2\omega_0)\right)\right) \end{aligned} \tag{A.62}$$

$$C_5 = 2 (q_0 - s)^4 \zeta^4 a_0' \beta_0^2 \left(1 - \eta^2\right)^{-7/2} \left(1 + \frac{11}{4}\eta(\eta + e_0) + e_0\eta^3\right) \tag{A.63}$$

$$D_2 = 4a_0' \zeta C_1^2 \tag{A.64}$$

$$D_3 = \frac{4}{3} a_0' \zeta^2 (17a_0' + s) C_1^3 \tag{A.65}$$

$$D_4 = \frac{2}{3} a_0' \zeta^3 (221a_0' + 31s) C_1^4 \tag{A.66}$$

The secular effects of the atmospheric drag and gravitation are included through the equations

$$M_{\mathrm{DF}} = M_0 + \left(1 + \frac{3J_2R_E^2\left(-1+3\Phi^2\right)}{4a_0'^2\beta_0^3} + \frac{3J_2^2R_E^4\left(13-78\Phi^2+137\Phi^4\right)}{64a_0'^4\beta_0^7}\right) n_0'\left(t-t_0\right) \tag{A.67}$$

$$\begin{aligned}\omega_{\mathrm{DF}} = {} & \omega_o + \left(-\frac{3J_2R_E^2\left(1-5\Phi^2\right)}{4a_0'^2\beta_0^4} + \frac{3J_2^2R_E^4\left(7-114\Phi^2+395\Phi^4\right)}{64a_0'^4\beta_0^8} - \right.\\ & \left.\frac{15J_4R_E^4\left(3-36\Phi^2+49\Phi^4\right)}{32a_0'^4\beta_0^8}\right) n_0'\left(t-t_0\right)\end{aligned} \tag{A.68}$$

$$\begin{aligned}\Omega_{\mathrm{DF}} = {} & \Omega_0 + \left(-\frac{3J_2R_E^2\Phi}{2a_0'^2\beta_0^4} + \frac{3J_2^2R_E^4\left(4\Phi-19\Phi^3\right)}{8a_0'^4\beta_0^8} - \right.\\ & \left.\frac{15J_4R_E^4\Phi\left(3-7\Phi^2\right)}{16a_0'^4\beta_0^8}\right) n_0'\left(t-t_0\right)\end{aligned} \tag{A.69}$$

$$\delta\omega = B^*C_3\cos\left(\omega_0\right)\left(t-t_0\right) \tag{A.70}$$

$$\delta M = -\frac{2}{3}\left(q_0-s\right)^4 B^*\zeta^4\frac{R_E}{e_0\eta}\left(\left(1+\eta\cos\left(M_{\mathrm{DF}}\right)\right)^3 - \left(1+\eta\cos\left(M_0\right)\right)^3\right) \tag{A.71}$$

$$M_p = M_{\mathrm{DF}} + \delta\omega + \delta M \tag{A.72}$$

$$\omega = \omega_{\mathrm{DF}} - \delta\omega - \delta M \tag{A.73}$$

$$\Omega = \Omega_{\mathrm{DF}} - \frac{21n_0'J_2R_E^2\Phi}{4a_0'^2\beta_0^2}C_1\left(t-t_0\right) \tag{A.74}$$

$$e = e_0 - B^*C_4\left(t-t_0\right) - B^*C_5\left(\sin\left(M_p\right)-\sin\left(M_0\right)\right) \tag{A.75}$$

$$a = a_0'\left(1 - C_1\left(t-t_0\right) - D_2\left(t-t_0\right)^2 - D_3\left(t-t_0\right)^3 - D_4\left(t-t_0\right)^4\right)^2 \tag{A.76}$$

$$\begin{aligned}L = {} & M_p + \omega + \Omega + n_0'\left(\frac{3}{2}C_1\left(t-t_0\right)^2 + \left(D_2+2C_1^2\right)\left(t-t_0\right)^3\right.\\ & +\frac{1}{4}\left(3D_3+12C_1D_2+10C_1^3\right)\left(t-t_0\right)^4 + \\ & \left.\frac{1}{5}\left(3D_4+12C_1D_3+6D_2^2+30C_1^2D_2+15C_1^4\right)\left(t-t_0\right)^5\right)\end{aligned} \tag{A.77}$$

$$\beta = \sqrt{1-e^2} \tag{A.78}$$

$$n = GM/a^{3/2} \tag{A.79}$$

where $(t-t_0)$ is the time since epoch. It should be noted that when epoch perigee height is less than 220 km, the equations for a and L are truncated after C_1 term and the terms involving C_5, $\delta\omega$ and δM are dropped. The long periodic terms are

$$a_{x\mathrm{N}} = e\cos\left(\omega\right) \tag{A.80}$$

$$L_L = -\frac{J_3R_E\sin\left(i_0\right)}{4J_2a\beta^2}\left(e\cos\left(\omega\right)\right)\left(\frac{3+5\Phi}{1+\Phi}\right) \tag{A.81}$$

$$a_{y\mathrm{NL}} = -\frac{J_3R_E\sin\left(i_0\right)}{2J_2a\beta^2} \tag{A.82}$$

$$L_T = L + L_L \tag{A.83}$$

$$a_{y\mathrm{N}} = e\sin\left(\omega\right) + a_{y\mathrm{NL}} \tag{A.84}$$

Then Kepler's equation $E+\omega$ has to be solved

$$\left(E+\omega\right)_{k+1} = \left(E+\omega\right)_k + \Delta\left(E+\omega\right)_k \tag{A.85}$$

with

$$(E+\omega)_1 = U = L_T - \Omega \tag{A.86}$$

$$\Delta\left(E+\omega\right)_k = \frac{U - a_{y\,\mathrm{N}}\cos\left(E+\omega\right)_k + a_{x\,\mathrm{N}}\sin\left(E+\omega\right)_k - \left(E+\omega\right)_k}{-a_{y\,\mathrm{N}}\sin\left(E+\omega\right)_k - a_{x\,\mathrm{N}}\cos\left(E+\omega\right)_k + 1} \tag{A.87}$$

The following equations are used to calculate preliminary quantities needed for short-period periodics (straight forward like described in section A.4.1)

$$e\cos\left(E\right) = a_{x\,\mathrm{N}}\,\cos\left(E+\omega\right) + a_{y\,\mathrm{N}}\,\sin\left(E+\omega\right) \tag{A.88}$$

$$e\sin\left(E\right) = a_{x\,\mathrm{N}}\,\sin\left(E+\omega\right) - a_{y\,\mathrm{N}}\,\cos\left(E+\omega\right) \tag{A.89}$$

$$e_L^2 = a_{x\,\mathrm{N}}^2 + a_{y\,\mathrm{N}}^2 \tag{A.90}$$

$$p_L = a\left(1 - e_L^2\right) \tag{A.91}$$

$$r = a\left(1 - e\cos\left(E\right)\right) \tag{A.92}$$

$$\dot{r} = GM\,\frac{\sqrt{a}}{r}\,e\sin\left(E\right) \tag{A.93}$$

$$r\dot{\nu} = GM\,\frac{\sqrt{p_L}}{r} \tag{A.94}$$

$$\sin\left(u\right) = \frac{a}{r}\left(\sin\left(E+\omega\right) - a_{y\,\mathrm{N}} - a_{x\,\mathrm{N}}\,\frac{e\sin\left(E\right)}{1+\sqrt{1-e_L^2}}\right) \tag{A.95}$$

$$\cos\left(u\right) = \frac{a}{r}\left(\cos\left(E+\omega\right) - a_{x\,\mathrm{N}} + a_{y\,\mathrm{N}}\,\frac{e\sin\left(E\right)}{1+\sqrt{1-e_L^2}}\right) \tag{A.96}$$

$$u = \tan^{-1}\left(\frac{\sin\left(u\right)}{\cos\left(u\right)}\right) \tag{A.97}$$

$$\Delta r = \frac{J_2 R_E^2}{4p_L}\left(1-\Phi^2\right)\cos\left(2u\right) \tag{A.98}$$

$$\Delta u = -\frac{J_2 R_E^2}{8p_L^2}\left(7\Phi^2-1\right)\sin\left(2u\right) \tag{A.99}$$

$$\Delta\Omega = \frac{3J_2 R_E^2\Phi}{4p_L^2}\sin\left(2u\right) \tag{A.100}$$

$$\Delta i = \frac{3J_2 R_E^2\Phi}{4p_L^2}\sin\left(i_0\right)\cos\left(2u\right) \tag{A.101}$$

$$\Delta\dot{r} = -\frac{J_2 R_E^2 n}{2p_L}\left(1-\Phi^2\right)\sin\left(2u\right) \tag{A.102}$$

$$\Delta r\dot{\nu} = \frac{J_2 R_E^2 n}{2p_L}\left(\left(1-\Phi^2\right)\cos\left(2u\right) - \frac{3}{2}\left(1-3\Phi^2\right)\right) \tag{A.103}$$

These short-period periodics are added to give the osculating quantities

$$r_k = r\left(1 - \frac{3}{4}J_2 R_E^2\frac{\sqrt{1-e_L^2}}{p_L^2}\left(3\Phi^2-1\right)\right) + \Delta r \tag{A.104}$$

$$u_k = u + \Delta u \tag{A.105}$$

$$\Omega_k = \Omega + \Delta\Omega \tag{A.106}$$

$$i_k = i_0 + \Delta i \tag{A.107}$$

$$\dot{r}_k = \dot{r} + \Delta\dot{r} \tag{A.108}$$

$$r\dot{\nu}_k = r\dot{\nu} + \Delta r\dot{\nu} \tag{A.109}$$

Then in accordance with the flow chart on page 17 the corrected unit orientation vectors could be calculated straight forward by substituting the sin and cos terms of ν by using u_k. Then position and velocity are given by

$$\vec{r} = r_k\vec{P} \qquad \text{and} \qquad \dot{\vec{r}} = \dot{r}\vec{P} + \left(r\dot{\nu}\right)_k\vec{Q} \tag{A.110}$$

It can be seen by consider these models, they have their basis in Newton's and Kepler's laws. The basic principles which are discovered by them are still valid, in order to get precise predictions it is just necessary to add new observations and measurements for the perturbation forces.

A.4.3 SDP4 Model

The SDP4 model is an extension of the SGP4 model for deep-space satellites (period greater than or equal 225 min). Here the original model of the Spacetrack Report No.3 is presented, nevertheless it should be mentioned, that the mathematics in the model is accurate but the implementation in the FORTRAN code has some errors which were corrected in the Spacetrack Report No.6 according to the analytical solutions in [5] and [29]. The corrected version is implemented in **Vis Sat**. This model implements also perturbations due to the effect that the satellite is longer above one area of the surface and the zonal specifications of that area have a bigger influence. Also it takes into account different celestial bodies.

$$\begin{aligned}
a_1 &= \left(\frac{GM}{n_0}\right)^{2/3} && \text{(A.111)}\\
\delta_1 &= \frac{3}{4}\,J_2\,\frac{R_E^2}{a_1^2}\,\frac{\left(3\cos^2(i_0)-1\right)}{\left(1-e_0^2\right)^{3/2}} && \text{(A.112)}\\
a_0 &= a_1\left(1-\frac{1}{3}\delta_1-\delta_1^2-\frac{134}{81}\delta_1^3\right) && \text{(A.113)}\\
\delta_0 &= \frac{3}{4}\,J_2\,\frac{R_E^2}{a_0^2}\,\frac{\left(3\cos^2(i_0)-1\right)}{\left(1-e_0^2\right)^{3/2}} && \text{(A.114)}\\
n_0' &= \frac{n_0}{1+\delta_0} && \text{(A.115)}\\
a_0' &= \frac{a_0}{1-\delta_0} && \text{(A.116)}\\
s^* &= \begin{cases} a_0'(1-e_0)-78+R_E & 98 < \text{ perigee } \leq 156\\ 20/6378.135+R_E & \text{perigee } \leq 98\end{cases} && \text{(A.117)}\\
q_0 &= a_0(1-e_0) && \text{(A.118)}
\end{aligned}$$

Then the constants can be calculated by using the appropriate s

$$\begin{aligned}
\Phi &= \cos(i_0) && \text{(A.119)}\\
\zeta &= \frac{1}{a_0'-s} && \text{(A.120)}\\
\beta_0 &= \left(1-e_0^2\right)^{1/2} && \text{(A.121)}\\
\eta &= a_0'e_0\zeta && \text{(A.122)}\\
C_2 &= (q_0-s)^4\zeta^4 n_0'\left(1-\eta^2\right)^{-7/2}\left(a_0'\left(1+\frac{3}{2}\eta^2+4e_0\eta+e_0\eta^3\right)+\right.\\
&\quad\left.+\frac{3}{4}\,J_2\frac{R_E^2\zeta}{(1-\eta^2)}\left(-\frac{1}{2}+\frac{3}{2}\Phi^2\right)\left(8+24\eta^2+3\eta^4\right)\right) && \text{(A.123)}\\
C_1 &= B^*C_2 && \text{(A.124)}\\
C_4 &= 2n_0'(q_0-s)^4\zeta^4 a_0'\beta_0^2\left(1-\eta^2\right)^{-7/2}\left(\left(2\eta(1+e_0\eta)+\frac{1}{2}e_0+\frac{1}{2}\eta^3\right)-\frac{J_2R_E^2}{a_0'(1-\eta^2)}\cdot\right.\\
&\quad\left(3\left(1-3\Phi^2\right)\left(1+\frac{3}{2}\eta^2-2e_0\eta-\frac{1}{2}e_0\eta^3\right)+\frac{3}{4}\left(1-\Phi^2\right)\right.\\
&\quad\left.\left.\left(2\eta^2-e_0\eta-e_0\eta^3\right)\cos(2\omega_0)\right)\right) && \text{(A.125)}
\end{aligned}$$

$$\dot{M} = \left(1 + \frac{3J_2R_E^2\left(-1+3\Phi^2\right)}{4a_0'^2\beta_0^3} + \frac{3J_2^2R_E^4\left(13-78\Phi^2+137\Phi^4\right)}{64a_0'^4\beta_0^7}\right)n_0' \tag{A.126}$$

$$\dot{\Omega}_1 = -\frac{3J_2R_E^2\Phi}{2a_0'^2\beta_0^4}n_0' \tag{A.127}$$

$$\dot{\omega} = \left(-\frac{3J_2R_E^2\left(1-5\Phi^2\right)}{4a_0'^2\beta_0^4} + \frac{3J_2^2R_E^4\left(7-114\Phi^2+395\Phi^4\right)}{64a_0'^4\beta_0^8} - \frac{15J_4R_E^4\left(3-36\Phi^2+49\Phi^4\right)}{32a_0'^4\beta_0^8}\right)n_0' \tag{A.128}$$

At this point SDP4 calls the initialisation section of the `DEEP` subroutine which calculates all initialised quantities needed for the deep-space perturbations. The secular effects of gravity are included by

$$M_{\mathrm{DF}} = M_0 + \dot{M}\left(t-t_0\right) \tag{A.129}$$

$$\omega_{\mathrm{DF}} = \omega_o + \dot{\omega}\left(t-t_0\right) \tag{A.130}$$

$$\Omega_{\mathrm{DF}} = \Omega_0 + \dot{\Omega}\left(t-t_0\right) \tag{A.131}$$

$$\tag{A.132}$$

where $(t-t_0)$ is time since epoch. The secular effect of drag on longitude of ascending node is included by

$$\Omega = \Omega_{\mathrm{DF}} - \frac{21}{4}\frac{n_0'J_2R_E^2\Phi}{a_0'^2\beta_0^2}C_1\left(t-t_0\right)^2 . \tag{A.133}$$

Next SDP4 calls the secular section of `DEEP` which adds the deep-space secular effects and long-period resonance effects to the classical Keplerian orbital elements. The secular effect of drag are included in the remaining elements by

$$a = a_{\mathrm{DS}}\left(1-C_1\left(t-t_0\right)\right)^2 \tag{A.134}$$

$$e = e_{\mathrm{DS}} - B^*C_4\left(t-t_0\right) \tag{A.135}$$

$$L = M_{\mathrm{DS}} + \omega_{\mathrm{DS}} + \Omega_{\mathrm{DS}} + n_0'\left(\frac{3}{2}C_1\left(t-t_0\right)^2\right) \tag{A.136}$$

where a_{DS}, e_{DS}, M_{DS}, ω_{DS} and Ω_{DS} are the values of n_0, e_0, M_{DF}, ω_{DS} and Ω after deep-space secular and resonance perturbations have been added.

Then SDP4 calls the periodics section of `DEEP` which adds the deep-space lunar and solar periodics to the orbital elements. From this points on it will be assumed that n, e, i, ω, Ω and M after lunar-solar periodics have been added.

Now the long term-period periodic terms are added. Up from this point the model is identical with the SGP4 model. The deep-space perturbations in `DEEP` are mainly developed from [21] and [22], they are extensive and should not be repeated here. How the function is integrated in the procedure of calculation is described above.

A.4.4 Boundaries of the NORAD models

The SGP models take into account that there are more forces acting on the satellite then just the gravitational force which is stated in Newton's law. In addition to these major force the other forces which causes additional accelerations on the satellite have to be taken into account in order to describe its trajectory. The SGP models were the first mathematically exact description of these forces. Since 1988 these models are valid and used in various kinds of propagation tools. Since these models were first introduced, they changed just in the precision of the TLE elements. In section 2.4.2 it was shown how different perturbations acts in which way on the Keplerian elements and in which way they change the orbit. Here very briefly the boundaries of these models should be shown for the sake of completeness.

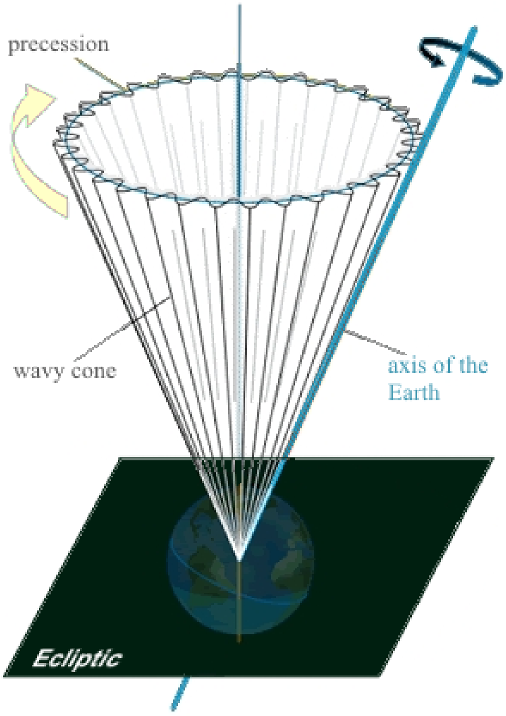


Figure A.2: Nutation

At first it should be mentioned that all the equations which are used in the SGP models do not contain relativistic effects due to the effect of the height and velocity. These principles are stated in Einstein's special theory of relativity. These effects are very minute but especially for time depending link establishment they are important (e.g. GPS system, due to the relativistic effects, the time on the satellites differ from UTC at about $5.28 \cdot 10^{-8}\%$. This error seems to be very small but for an orbital period, this is a differenece of $4.44 \cdot 10^{-10}T$ and leads to an error in the orbit of about 13.3 cm per period).

Another effect which is not included in the SGP model is the nutation of the Earth. The Earth is a non ideal spherical body (cf. fig. A.1) especially due to the flattening of the Earth, the equatorial diameter is ≈ 43 km bigger than the polar diameter. The Sun and the Moon acts on this oblateness. This force is bigger on the day side as it is on the night sight, therefore there is a torque which tries to stabilises the axis of the Earth. This means this force tries to reduce the tilt of the ecliptic. In this case the tilt of the Earth's axis goes around in a circle in the opposite direction to the spinning direction, the precession is considered in the SGP models. The nutation is a superposed motion to the precession see fig. A.2. This force is a result of the Moon's orbit. The orbit of the Moon is inclined by 5.1° to the ecliptic, thus that the declination is changing every time. Due to this inclined orbit the gravitational force of the Moon has to be added or subtracted from the Sun's gravitational effect. But the effect of this nutation is very small the semimajor axis of this nutation ellipse is $9''.6$ across the precession and the semiminor axis is $6''.9$ in the direction of the precession. Therefore this motion has just a minute effect on the orbit of a satellite.

The SGP models also does not consider forces due to the gravitational effects of other planets in our solar system. In order to do so the three body or even the n body problem has to be considered. The SDP4 model tries to solve this problem in a numerical way but up to now this is just an approximation to the solution of a n body problem. Also the other perturbations in deep space are just considered as mean forces which acts on the satellite without precise analytical solutions.

A.5 Acceleration on Earth orbiting Satellites

In addition to the perturbations which were discussed in the sections above, this section gives an overview of all additional accelerations which could have an influence on the orbit of a satellite. Some of these forces could only be derived in very complex numerical models and should not be considered here. For more details see [33].

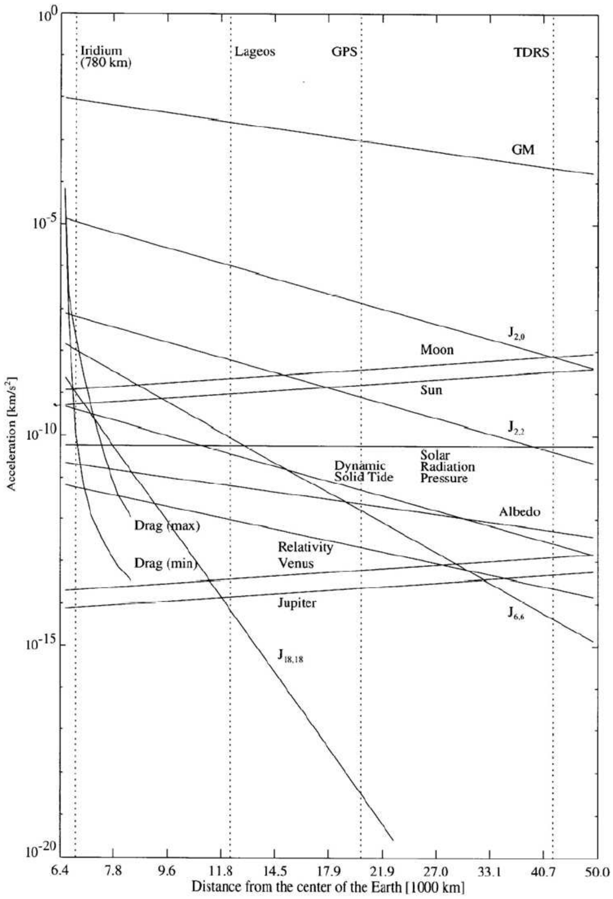


Figure A.3: Accelerations on Earth orbiting Satellites (source: [33])

A.6 Earth's Magnetic Field Additions

A.6.1 Quasi Schmidt normalised

The geomagnetic scalar potential V has quasi Schmidt- normalised associated Legendre functions in the expression $\vec{P}_n^m \cos(\theta)$ of degree n and order m. They can be described by the associated Legendre functions which are normalised by the constants

$$K_n = \begin{cases} \frac{1}{2^n} & m = 0 \\ \\ \frac{1}{2^n}\sqrt{\frac{2(n-m)!}{(n+m)!}} & m > 0 \end{cases} \tag{A.137}$$

which gives the normalised associated Legendre functions

$$P_n^m = \begin{cases} \left(\frac{1}{2^n n!}\right)\left(\frac{\partial}{\partial x}\right)^n \left(x^2-1\right)^n & m = 0 \\ \sqrt{\frac{2(n-m)!}{(n+m)!}\left(1-x^2\right)^m}\left(\frac{1}{2^n n!}\right)\left(\frac{\partial}{\partial x}\right)^n & m > 0 \end{cases} \tag{A.138}$$

B MATLAB

This section shows the MATLAB® functions which are included in **VIS SAT** and some additional information of the implementation. The functions and the HTML documentation is included on the CD-Rom.

B.1 Explanation: MATLAB® /`MEX`

`MEX` functions (or files) are executable C/C++ or FORTRAN subroutines in the MATLAB® environment which are produced by the specified `mex`-compiler. The compiler generates under Windows© an `dll` file which can be used under MATLAB® like a normal built in function. The advantage of such implementations are, that the functions can use the large C/C++ or FORTRAN libraries during the calculations. Especially when the code contains a huge amount of loop structures or scalar mathematical operations it is reasonable to implement this code in a `MEX` function, because MATLAB® has a weakness when it comes to loop structures, so in order to keep the processing time to a minimum a `MEX` function is the best choice[9]. In MATLAB® the command `mex` creates from a C/C++ code the `dll` file. In order to do so, the C/C++ code must contain a gateway function called `mexFunction()` which connects the MATLAB® environment with the C/C++ code. The usage messages must be described in an additional `m`-file with the same name as the `MEX` function. Then MATLAB® will show the content of the `m`-file after the command `>> help <name of dll>`. The gateway function of the `MEX` function has the following syntax

```
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray *prhs[])
```

The four parameter describes the input and output arguments of the `MEX` function (left hand side and right hand side). The field `plhs` and `prhs` are pointer for the objects, `nlhs` and `nrhs` contain the number of elements in the field. In the C/C++ code it is possible to access these fields by using the function `void getParameters(...)`. So it is possible to set the local variables in the C/C++ code to the passed variables in MATLAB® .

B.2 `norad2kep.m`

```
function satellite=norad2kep(file,name)
%NORAD2KEP  Get orbit elements from NORAD two-line element data, does not
% include the classification data. The TLE data could be received from
% http://celestrak.com/NORAD/elements/
%
%  SATELLITE = NORAD2KEP([FILE],[NAME]), is an array of struct
%
%  SATELLITE.name             name of the satellite
```

---

[9] It should be mentioned here, that there are some incompatibilities in the MATLAB® 7.0 SP1 environment for FORTRAN `MEX` functions see also http://www.mathworks.com/support/tech-notes/1600/1601_files/1601_70.html thats why in this thesis a conversion from FORTRAN to C++ was necessary before using a `MEX` function and only the C++ `MEX` procedure should be described here

```
%  SATELLITE.tle             tle elements
%  SATELLITE.epochC          calender date
%  SATELLITE.epochN          NORAD date
%
%  Structure of tle
%  tle(1) Julian date, tle(2) 1st derivation n0, tle(3) 2nd derivation n0
%  tle(4) BSTAR, tle(5) inclination, tle(6) RAAN, tle(7) eccentricity,
%  tle(8) argument of perigee, tle(9) mean anomaly, tle(10) mean motion
%  tle(11) revolution number;
%
%  notation/conversions see:
%  Davidoff, M.R., Satellite Experimenters Handbook, 2nd Edition, American
%  Radio Relay League, 1990.
%
% Copyright (c) 2005-08-20, P.Daum
% the NORAD data elemets are included in the VIS SAT 1.0 package


% ------------------------- begin of open file -------------------------
% default input control
if nargin<2
  msgbox('no satellite name specified','info','warn');
  satellite=[];  % no information
  return;
end;
if nargin==0
  msgbox('no filename specified','info','warn');
  satellite=[];  % no information
  return;
end;
if exist(file,'file')==0
  msgbox('file does not exist','info','warn');
  satellite=[];  % no information
  return;
end;

% open specified file
fp=fopen(file,'r');

% serach for satellites name
found=false;
while (feof(fp) == 0)
  % read three lines of each TLE record
  cline1 = fgetl(fp);
  cline2 = fgetl(fp);
  cline3 = fgetl(fp);
  % look for specified satellite name
  if (strcmp(lower(deblank(cline1)), lower(deblank(name)))==1)
     found=true; break;
  end
end

% specified satellite is not found
if not(found)
  msgbox(['satellite with the name ',upper(name),' not found'],...
      'info','warn');
  satellite=[];  % no information
  return;
end;

fclose(fp);
% ------------------------- end of open file ---------------------------
ae = 1;

% epoch year (2 digits)
yr = str2num(cline2(19:20));
% epoch day fraction
dayofyear = str2num(cline2(21:32));
% NORAD specification
epochN=(1e3.*yr)+ dayofyear;
% time conversion
if (yr < 57) yr = 2000 + yr; else yr = 1900 + yr; end
xjdtmp = julian(1, 0, yr);
% Julian date of the epoch
xjdtle = xjdtmp + dayofyear;

% d/dt of mean motin
ndt = str2num(cline2(34:43));
% d2/dt2 of mean motin
nd2t = 1.0e-5.*str2num(cline2(45:50)).*(10^str2num(cline2(51:52)));
% bstar drag term
bstar = 1.0e-5.*str2num(cline2(54:59)).*(10^str2num(cline2(60:61)))./ae;
% orbit inclination
incl = str2num(cline3(9:16));
% right ascension of ascending node
raan = str2num(cline3(18:25));
% eccentricity
ecc = 1.0e-7.*str2num(cline3(27:33));
% argument of perigee
omegao = str2num(cline3(35:42));
% mean anomaly
mo = str2num(cline3(44:51));
% mean motion
no = str2num(cline3(53:63));
% revolution number
rev = str2num(cline3(64:68));

% calender date conversion
%                    year,  month,  day/time
```

```
epoch=datevec(datenum(yr,    0  ,  dayofyear));
% could be converted to normal calender date by datestr(epoch,0)

% return of the struct
satellite.name=deblank(upper(name));
% orbit elements
satellite.tle(1)=xjdtle;
satellite.tle(2)=ndt;
satellite.tle(3)=nd2t;
satellite.tle(4)=bstar;
satellite.tle(5)=incl;
satellite.tle(6)=raan;
satellite.tle(7)=ecc;
satellite.tle(8)=omegao;
satellite.tle(9)=mo;
satellite.tle(10)=no;
satellite.tle(11)=rev;
% epoch in calender date for
satellite.epochC=epoch;
% epoch in special date form
satellite.epochN=epochN;
return

% ------------------------------------------------------------------------
% ------------------- Julian date conversion -----------------------------
% ------------------------------------------------------------------------
function jdate = julian (month, day, year)
y = year; m = month; b = 0; c = 0;
if (m <= 2)
   y = y - 1;
   m = m + 12;
end
if (y < 0)
   c = -.75;
end
% check for valid calendar date
if (year < 1582)
   % null
elseif (year > 1582)
   a = fix(y / 100);
   b = 2 - a + floor(a / 4);
elseif (month < 10)
   % null
elseif (month > 10)
   a = fix(y / 100);
   b = 2 - a + floor(a / 4);
elseif (day <= 4)
   % null
elseif (day > 14)
   a = fix(y / 100);
   b = 2 - a + floor(a / 4);
else
  msgbox('this is an invalid calendar date','info','warn');
  jdate=[];  % no information
  return;
end

% calculate the Julian date
jd = fix(365.25 * y + c) + fix(30.6001 * (m + 1));
jdate = jd + day + b + 1720994.5;
return;
```

B.3 `orbit.m`

```
/***********************************************************************************************************************************\
*                                           MATLAB MEX FUNCTION ORBIT                                                               *
*                                      SGP4/SDP4 implementation for VIS SAT                                                         *
\***********************************************************************************************************************************/

/*---------------------------------------------------------------------------------------------------------------------------------*/
/*********************************************** DEFINITION PART *******************************************************************/
/*---------------------------------------------------------------------------------------------------------------------------------*/
#define _ISOC99_SOURCE

#include <ctype.h>
#include <curses.h>
#include <fcntl.h>
#include <math.h>
#include <netdb.h>
#include <netinet/in.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <termios.h>
#include <time.h>
#include <unistd.h>

#include "mex.h"
#include "matrix.h"


/* constants used by SGP4/SDP4 code                                          */
/* direct imported from FORTRAN                                              */
#define deg2rad   1.745329251994330E-2     /* degrees to radians */
#define pi        3.14159265358979323846   /* PI     */
```

```
#define pio2      1.57079632679489656      /* PI/2   */
#define twopi     6.28318530717958623      /* 2*PI   */
#define x3pio2    4.71238898038468967      /* 3*PI/2 */
#define e6a       1.0E-6
#define tothrd    6.6666666666666666E-1    /* 2/3 */
#define xj2       1.0826158E-3             /* J2 harmonic */
#define xj3       -2.53881E-6              /* J3 harmonic */
#define xj4       -1.65597E-6              /* J4 harmonic */
#define xke       7.43669161E-2
#define xkmper    6.378137E3               /* Earth radius km  */
#define xmnpda    1.44E3                   /* minutes per day  */
#define ae        1.0
#define ck2       5.413079E-4
#define ck4       6.209887E-7
#define f         3.35281066474748E-3      /* flattening factor    */
#define ge        3.986008E5               /* Earth grav. constant */
#define s         1.012229
#define qoms2t    1.880279E-09
#define secday    8.6400E4                 /* seconds per day      */
#define omega_E   1.00273790934            /* Earth rot./sid. day  */
#define omega_ER  6.3003879                /* Earth rot. rads/sid. */
#define zns       1.19459E-5
#define c1ss      2.9864797E-6
#define zes       1.675E-2
#define znl       1.5835218E-4
#define c1l       4.7968065E-7
#define zel       5.490E-2
#define zcosis    9.1744867E-1
#define zsinis    3.9785416E-1
#define zsings   -9.8088458E-1
#define zcosgs    1.945905E-1
#define zcoshs    1
#define zsinhs    0
#define q22       1.7891679E-6
#define q31       2.1460748E-6
#define q33       2.2123015E-7
#define g22       5.7686396
#define g32       9.5240898E-1
#define g44       1.8014998
#define g52       1.0508330
#define g54       4.4108898
#define root22    1.7891679E-6
#define root32    3.7393792E-7
#define root44    7.3636953E-9
#define root52    1.1428639E-7
#define root54    2.1765803E-9
#define thdt      4.3752691E-3
#define rho       1.5696615E-1
#define sr        6.96000E5                 /* solar radius */


/* entry points of DEEP for SDP4                                             */
#define dpinit   1  /* Deep-space initialization code   */
#define dpsec    2  /* Deep-space secular code          */
#define dpper    3  /* Deep-space periodic code         */

/* the flow is controlled by flags, to displace the GO TO in FORTRAN         */
#define ALL_FLAGS            -1
#define SGP4_INITIALIZED_FLAG 0x000002
#define SDP4_INITIALIZED_FLAG 0x000004
#define SIMPLE_FLAG           0x000020
#define DEEP_SPACE_EPHEM_FLAG 0x000040
#define LUNAR_TERMS_DONE_FLAG 0x000080
#define DO_LOOP_FLAG          0x000200
#define RESONANCE_FLAG        0x000400
#define SYNCHRONOUS_FLAG      0x000800
#define EPOCH_RESTART_FLAG    0x001000

/* Two-line-element satellite orbital data structure                         */
typedef struct {
   double  epoch, xndt2o, xndd6o, bstar, xincl,
       xnodeo, eo, omegao, xmo, xno;
     int    catnr, elset, revnum;
     char    sat_name[25], idesg[9];
}  tle_t;

/* geodetic position structure used by SGP4/SDP4 code.                       */
typedef struct {
   double lat, lon, alt, theta;
}  geodetic_t;

/* general three-dimensional vector structure used by SGP4/SDP4 code.        */
typedef struct {
   double x, y, z, w;
}  vector_t;

/* common arguments between deep-space functions used by SGP4/SDP4 code.     */
typedef struct {
       /* Used by dpinit part of DEEP */
   double  eosq, sinio, cosio, betao, aodp, theta2,
   sing, cosg, betao2, xmdot, omgdot, xnodot, xnodp;

   /* Used by dpsec and dpper parts of DEEP */
   double  xll, omgadf, xnode, em, xinc, xn, t;

     /* Used by thetg and DEEP */
   double  ds50;
}  deep_arg_t;
```

```
/*----------------------------------------------------------------------------------------------------------------------------------*/
/*********************************************** END DEFINITION PART ****************************************************************/
/*----------------------------------------------------------------------------------------------------------------------------------*/


/*----------------------------------------------------------------------------------------------------------------------------------*/
/************************************************** INTERFACES **********************************************************************/
/*----------------------------------------------------------------------------------------------------------------------------------*/
/*  MATLAB gateway                                                             */
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]);
void getParameters( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]);

/*  FLAG functions                                                             */
int isFlagSet(int flag);
int isFlagClear(int flag);
void SetFlag(int flag);
void ClearFlag(int flag);

/*  MATHEMATICAL functions                                                     */
int Sign(double arg);
double Sqr(double arg);
double Cube(double arg);
double Radians(double arg);
double Degrees(double arg);
double ArcSin(double arg);
double ArcCos(double arg);
void Magnitude(vector_t *v);
void Vec_Add(vector_t *v1, vector_t *v2, vector_t *v3);
void Vec_Sub(vector_t *v1, vector_t *v2, vector_t *v3);
void Scalar_Multiply(double k, vector_t *v1, vector_t *v2);
void Scale_Vector(double k, vector_t *v);
double Dot(vector_t *v1, vector_t *v2);
double Angle(vector_t *v1, vector_t *v2);
void Cross(vector_t *v1, vector_t *v2 ,vector_t *v3);
void Normalize(vector_t *v);
double AcTan(double sinx, double cosx);
double FMod2p(double x);
double FixAngle(double x);
double Modulus(double arg1, double arg2);
double Frac(double arg);
int Round(double arg);
double Int(double arg);

int Sat_Eclipsed(vector_t *pos, vector_t *sol, double *depth);


/* time conversion                                                             */
void Date_Time(double julian_date, struct tm *cdate);
double Julian_Date(struct tm *cdate);
double Fraction_of_Day(int hr, int mi, double se);
int DOY (int yr, int mo, int dy);
double Julian_Date_of_Epoch(double epoch);
double Julian_Date_of_Year(double year);
double ThetaG_JD(double jd);
double ThetaG(double epoch, deep_arg_t *deep_arg);

/* SGP4/SDP4 functions                                                         */
void select_ephemeris(tle_t *tle);
void SGP4(double tsince, tle_t * tle, double *position, double *velocity);
void Deep(int ientry, tle_t * tle, deep_arg_t * deep_arg);
void SDP4(double tsince, tle_t * tle, double *position, double *velocity);

/* global variables                                                            */
double tsince, jul_epoch, jul_utc, eclipse_depth=0,
    phase, daynum, ax, ay, az, rx, ry, rz, squint, alat, alon;


char temp[80], calc_squint;

int Flags=0;

/* global structure used by SGP4/SDP4 code                                     */
geodetic_t obs_geodetic;

/* global TLE for the satellite used by SGP4/SDP4 code.                        */
tle_t tle;


/*----------------------------------------------------------------------------------------------------------------------------------*/
/******************************************************** GATEWAY *******************************************************************/
/*----------------------------------------------------------------------------------------------------------------------------------*/
void mexFunction( int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
{
  ClearFlag(ALL_FLAGS);
  getParameters(nlhs, plhs, nrhs, prhs);
  select_ephemeris(&tle);

  if (isFlagSet(DEEP_SPACE_EPHEM_FLAG))
SDP4(tsince, &tle, mxGetPr(plhs[0]),mxGetPr(plhs[1]));
else
SGP4(tsince, &tle, mxGetPr(plhs[0]),mxGetPr(plhs[1]));
}
/*----------------------------------------------------------------------------------------------------------------------------------*/
/******************************************************* END GATEWAY ****************************************************************/
/*----------------------------------------------------------------------------------------------------------------------------------*/


/*
 * Set default values and create vectors for output.
 *
 * SYNTAX
 *    orbit(xjdtle,ndt,nd2t,bstar,incl,raan,ecc,omegao,mo,no,rev,tsince)
 */
void getParameters( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
  /* input parameters */
tle.epoch= *mxGetPr(prhs[0]);
tle.xndt2o= *mxGetPr(prhs[1]);
tle.xndd6o= *mxGetPr(prhs[2]);
tle.bstar= *mxGetPr(prhs[3]);
tle.xincl= *mxGetPr(prhs[4]);
tle.xnodeo= *mxGetPr(prhs[5]);
tle.eo= *mxGetPr(prhs[6]);
tle.omegao= *mxGetPr(prhs[7]);
tle.xmo= *mxGetPr(prhs[8]);
tle.xno=              *mxGetPr(prhs[9]);
tle.revnum= *mxGetPr(prhs[10]);
tsince =              *mxGetPr(prhs[11]);


  /* output vectors */
plhs[0]=mxCreateDoubleMatrix(3, 1, mxREAL); /* position vector 3x1 real */
plhs[1]=mxCreateDoubleMatrix(3, 1, mxREAL); /* velocity vector 3x1 real */
}

/* testing and setting/clearing flags used in SGP4/SDP4 code                 */
int isFlagSet(int flag)
{
return (Flags&flag);
}

int isFlagClear(int flag)
{
return (~Flags&flag);
}

void SetFlag(int flag)
{
Flags|=flag;
}

void ClearFlag(int flag)
{
Flags&=~flag;
}


/*-----------------------------------------------------------------------------------------------------------------------------*/
/* functions for SGP4/SDP4                                                                                                     */
/*-----------------------------------------------------------------------------------------------------------------------------*/


/* returns sign of a double                                                  */
int Sign(double arg)
{
if (arg>0)
return 1;

else if (arg<0)
return -1;

else
return 0;
}

/* returns square of a double                                                */
double Sqr(double arg)
{
return (arg*arg);
}

/* returns cube of a double                                                  */
double Cube(double arg)
{
return (arg*arg*arg);
}

/* returns angle in radians from argument in degrees                         */
double Radians(double arg)
{
return (arg*deg2rad);
}

/* returns angle in degrees from argument in radians                         */
double Degrees(double arg)
{
return (arg/deg2rad);
}

/* returns the arcsine of the argument                                       */
double ArcSin(double arg)
{
if (fabs(arg)>=1.0)
return(Sign(arg)*pio2);
else

return(atan(arg/sqrt(1.0-arg*arg)));
}
```

```
/* returns arccosine of argument                                          */
double ArcCos(double arg)
{
return(pio2-ArcSin(arg));
}

/* calculates scalar magnitude of a vector_t argument                     */
void Magnitude(vector_t *v)
{
v->w=sqrt(Sqr(v->x)+Sqr(v->y)+Sqr(v->z));
}

/* adds vectors v1 and v2 together to produce v3                          */
void Vec_Add(vector_t *v1, vector_t *v2, vector_t *v3)
{
v3->x=v1->x+v2->x;
v3->y=v1->y+v2->y;
v3->z=v1->z+v2->z;
Magnitude(v3);
}

/* subtracts vector v2 from v1 to produce v3                              */
void Vec_Sub(vector_t *v1, vector_t *v2, vector_t *v3)
{
v3->x=v1->x-v2->x;
v3->y=v1->y-v2->y;
v3->z=v1->z-v2->z;
Magnitude(v3);
}

/* multiplies the vector v1 by the scalar k to produce the vector v2        */
void Scalar_Multiply(double k, vector_t *v1, vector_t *v2)
{
v2->x=k*v1->x;
v2->y=k*v1->y;
v2->z=k*v1->z;
v2->w=fabs(k)*v1->w;
}

/* multiplies the vector v1 by the scalar k                               */
void Scale_Vector(double k, vector_t *v)
{
v->x*=k;
v->y*=k;
v->z*=k;
Magnitude(v);
}

/* returns the dot product of two vectors                                 */
double Dot(vector_t *v1, vector_t *v2)
{
return (v1->x*v2->x+v1->y*v2->y+v1->z*v2->z);
}

/* calculates the angle between vectors v1 and v2                         */
double Angle(vector_t *v1, vector_t *v2)
{
Magnitude(v1);
Magnitude(v2);
return(ArcCos(Dot(v1,v2)/(v1->w*v2->w)));
}

/* produces cross product of v1 and v2, and returns in v3                 */
void Cross(vector_t *v1, vector_t *v2 ,vector_t *v3)
{
v3->x=v1->y*v2->z-v1->z*v2->y;
v3->y=v1->z*v2->x-v1->x*v2->z;
v3->z=v1->x*v2->y-v1->y*v2->x;
Magnitude(v3);
}

/* mormalizes a vector                                                    */
void Normalize(vector_t *v)
{
v->x/=v->w;
v->y/=v->w;
v->z/=v->w;
}

/* four-quadrant arctan function                                          */
double AcTan(double sinx, double cosx)
{
if (cosx==0.0)
{
if (sinx>0.0)
return (pio2);
else
return (x3pio2);
}

else
{
if (cosx>0.0)
{
if (sinx>0.0)
return (atan(sinx/cosx));
else
return (twopi+atan(sinx/cosx));
}

else
return (pi+atan(sinx/cosx));
}
}

/* returns mod 2PI of argument                                         */
double FMod2p(double x)
{
int i;
double ret_val;

ret_val=x;
i=ret_val/twopi;
ret_val-=i*twopi;

if (ret_val<0.0)
ret_val+=twopi;

return ret_val;
}


/* reduces angles greater than two pi by subtracting two pi from the angle   */
double FixAngle(x)
double x;
{
while (x>twopi)
x-=twopi;

return x;
}

/* returns arg1 mod arg2                                               */
double Modulus(double arg1, double arg2)
{
int i;
double ret_val;

ret_val=arg1;
i=ret_val/arg2;
ret_val-=i*arg2;

if (ret_val<0.0)
ret_val+=arg2;

return ret_val;
}

/* returns fractional part of double argument                          */
double Frac(double arg)
{
return(arg-floor(arg));
}

/* returns argument rounded up to nearest integer                      */
int Round(double arg)
{
return((int)floor(arg+0.5));
}

/* returns the floor integer of a double arguement, as double          */
double Int(double arg)
{
return(floor(arg));
}

/* calculates the Julian Date                                          */
double Julian_Date_of_Year(double year)
{
/* Astronomical Formulae for Calculators, Jean Meeus,  */
/* pages 23-25. Calculate Julian Date of 0.0 Jan year  */

long A, B, i;
double jdoy;

year=year-1;
i=year/100;
A=i;
i=A/4;
B=2-A+i;
i=365.25*year;
i+=30.6001*14;
jdoy=i+1720994.5+B;

return jdoy;
}

/* returns the Julian Date of an epoch specified in the format used in      */
/* the NORAD two-line element sets.                                          */
double Julian_Date_of_Epoch(double epoch)
{
double year, day;

/* modification to support Y2K */
/* valid 1957 through 2056     */

day=modf(epoch*1E-3, &year)*1E3;
```

```
if (year<57)
year=year+2000;
else
year=year+1900;

return (Julian_Date_of_Year(year)+day);
}

/* calculates the day of the year for the specified date. The calculation    */
/* uses the rules for the Gregorian calendar                                 */
int DOY (int yr, int mo, int dy)
{
const int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
int i, day;

day=0;

for (i=0; i<mo-1; i++ )
    day+=days[i];

day=day+dy;

/* Leap year correction */

if ((yr%4==0) && ((yr%100!=0) || (yr%400==0)) && (mo>2))
day++;

return day;
}

/* calculates the fraction of a day passed at the specified input time.      */
double Fraction_of_Day(int hr, int mi, double se)
{
double dhr, dmi;

dhr=(double)hr;
dmi=(double)mi;

return ((dhr+(dmi+se/60.0)/60.0)/24.0);
}

/* converts a standard calendar date and time to a Julian Date.              */
double Julian_Date(struct tm *cdate)
{
double julian_date;

julian_date=Julian_Date_of_Year(cdate->tm_year)+DOY(cdate->tm_year,
cdate->tm_mon,cdate->tm_mday)+Fraction_of_Day(cdate->tm_hour,
  cdate->tm_min,cdate->tm_sec)+5.787037e-06;

return julian_date;
}


/* converts a Julian Date to standard calendar date and time                 */
void Date_Time(double julian_date, struct tm *cdate)
{
time_t jtime;

jtime=(julian_date-2440587.5)*86400.0;
*cdate=*gmtime(&jtime);
}

/* The function ThetaG_JD calculates the Greenwich Mean Sidereal Time for an */
/* epoch specified in the JD                                                */

/* Reference:  The 1992 Astronomical Almanac, page B6.                       */
double ThetaG_JD(double jd)
{
double UT, TU, GMST;

UT=Frac(jd+0.5);
jd=jd-UT;
TU=(jd-2451545.0)/36525;
GMST=24110.54841+TU*(8640184.812866+TU*(0.093104-TU*6.2E-6));
GMST=Modulus(GMST+secday*omega_E*UT,secday);

return (twopi*GMST/secday);
}


/* The function ThetaG calculates the Greenwich Mean Sidereal Time  for an   */
/* epoch specified in the format used in the NORAD two-line element sets.    */

/* Reference:  The 1992 Astronomical Almanac, page B6.                       */
double ThetaG(double epoch, deep_arg_t *deep_arg)
{
double year, day, UT, jd, TU, GMST, ThetaG;

/* Modification to support Y2K */
/* Valid 1957 through 2056     */

day=modf(epoch*1E-3,&year)*1E3;

if (year<57)
year+=2000;
else
year+=1900;
```

```
UT=modf(day,&day);
jd=Julian_Date_of_Year(year)+day;
TU=(jd-2451545.0)/36525;
GMST=24110.54841+TU*(8640184.812866+TU*(0.093104-TU*6.2E-6));
GMST=Modulus(GMST+secday*omega_E*UT,secday);
ThetaG=twopi*GMST/secday;
deep_arg->ds50=jd-2433281.5+UT;
ThetaG=FMod2p(6.3003880987*deep_arg->ds50+1.72944494);

return ThetaG;
}

/* calculates satellite’s eclipse status and depth                           */
int Sat_Eclipsed(vector_t *pos, vector_t *sol, double *depth)
{
double sd_sun, sd_earth, delta;
vector_t Rho, earth;

/* determine partial eclipse */
sd_earth=ArcSin(xkmper/pos->w);
Vec_Sub(sol,pos,&Rho);
sd_sun=ArcSin(sr/Rho.w);
Scalar_Multiply(-1,pos,&earth);
delta=Angle(sol,&earth);
*depth=sd_earth-sd_sun-delta;

if (sd_earth<sd_sun)
return 0;
else
if (*depth>=0)
return 1;
else
return 0;
}
/*--------------------------------------------------------------------------------------------------------------------------------*/
/* end functions for SGP4/SDP4                                                                                                    */
/*--------------------------------------------------------------------------------------------------------------------------------*/

/*--------------------------------------------------------------------------------------------------------------------------------*/
/* SGP4/SDP4 implementation                                                                                                       */
/*--------------------------------------------------------------------------------------------------------------------------------*/

/* Selects the apropriate ephemeris type to be used for predictions          */
/* according to the data in the TLE, it also processes values in the tle set */
/* so that they are apropriate for the SGP4/SDP4 routines                    */
void select_ephemeris(tle_t *tle)
{
double ao, xnodp, dd1, dd2, delo, temp, a1, del1, r1;

/* preprocess tle set */
tle->xnodeo*=deg2rad;
tle->omegao*=deg2rad;
tle->xmo*=deg2rad;
tle->xincl*=deg2rad;
temp=twopi/xmnpda/xmnpda;
tle->xno=tle->xno*temp*xmnpda;
tle->xndt2o*=temp;
tle->xndd6o=tle->xndd6o*temp/xmnpda;
tle->bstar/=ae;

/* period > 225 minutes is deep space */
dd1=(xke/tle->xno);
dd2=tothrd;
a1=pow(dd1,dd2);
r1=cos(tle->xincl);
dd1=(1.0-tle->eo*tle->eo);
temp=ck2*1.5f*(r1*r1*3.0-1.0)*pow(dd1,-1.5);
del1=temp/(a1*a1);
ao=a1*(1.0-del1*(tothrd*.5+del1*(del1*1.654320987654321+1.0)));
delo=temp/(ao*ao);
xnodp=tle->xno/(delo+1.0);

/* select a deep-space/near-earth ephemeris */
if (twopi/xnodp/xmnpda>=0.15625)
SetFlag(DEEP_SPACE_EPHEM_FLAG);
else
ClearFlag(DEEP_SPACE_EPHEM_FLAG);
}


/*--------------------------------------------------------------------------------------------------------------------------------*/
/**************************************** SGP4 ************************************************************************************/
/*--------------------------------------------------------------------------------------------------------------------------------*/

/* SGP 4 implementation. tsince is time since epoch in minutes, tle is a     */
/* pointer to a tle_t structure with Keplerian orbital elements and position */
/* and velocity are the returning ECI satellite position and                 */
/* velocity in m and m/s.                                                    */
void SGP4(double tsince, tle_t * tle, double *position, double *velocity)
{
static double aodp, aycof, c1, c4, c5, cosio, d2, d3, d4, delmo,
omgcof, eta, omgdot, sinio, xnodp, sinmo, t2cof, t3cof, t4cof,
t5cof, x1mth2, x3thm1, x7thm1, xmcof, xmdot, xnodcf, xnodot, xlcof;

double cosuk, sinuk, rfdotk, vx, vy, vz, ux, uy, uz, xmy, xmx, cosnok,
sinnok, cosik, sinik, rdotk, xinck, xnodek, uk, rk, cos2u, sin2u,
u, sinu, cosu, betal, rfdot, rdot, r, pl, elsq, esine, ecose, epw,
cosepw, x1m5th, xhdot1, tfour, sinepw, capu, ayn, xlt, aynl, xll,
axn, xn, beta, xl, e, a, tcube, delm, delomg, templ, tempe, tempa,
xnode, tsq, xmp, omega, xnoddf, omgadf, xmdf, a1, a3ovk2, ao,
betao, betao2, c1sq, c2, c3, coef, coef1, del1, delo, eeta, eosq,
etasq, perigee, pinvsq, psisq, qoms24, s4, temp, temp1, temp2,
temp3, temp4, temp5, temp6, theta2, theta4, tsi;

int i;

/* initialisation */
if (isFlagClear(SGP4_INITIALIZED_FLAG))
{
SetFlag(SGP4_INITIALIZED_FLAG);

/* recover original mean motion (xnodp) and   */
/* semimajor axis (aodp) from input elements. */
a1=pow(xke/tle->xno,tothrd);
cosio=cos(tle->xincl);
theta2=cosio*cosio;
x3thm1=3*theta2-1.0;
eosq=tle->eo*tle->eo;
betao2=1.0-eosq;
betao=sqrt(betao2);
del1=1.5*ck2*x3thm1/(a1*a1*betao*betao2);
ao=a1*(1.0-del1*(0.5*tothrd+del1*(1.0+134.0/81.0*del1)));
delo=1.5*ck2*x3thm1/(ao*ao*betao*betao2);
xnodp=tle->xno/(1.0+delo);
aodp=ao/(1.0-delo);

/* for perigee less than 220 kilometers, the "simple"     */
/* flag is set and the equations are truncated to linear  */
/* variation in sqrt a and quadratic variation in mean    */
/* anomaly.  Also, the c3 term, the delta omega term, and */
/* the delta m term are dropped. (see thesis appendix A.4)*/
if ((aodp*(1-tle->eo)/ae)<(220/xkmper+ae))
    SetFlag(SIMPLE_FLAG);

else
    ClearFlag(SIMPLE_FLAG);

/* for perigees below 156 km, the      */
/* values of s and qoms2t are altered. */
s4=s;
qoms24=qoms2t;
perigee=(aodp*(1-tle->eo)-ae)*xkmper;

if (perigee<156.0)
{
if (perigee<=98.0)
    s4=20;
else
     s4=perigee-78.0;

qoms24=pow((120-s4)*ae/xkmper,4);
s4=s4/xkmper+ae;
}

pinvsq=1/(aodp*aodp*betao2*betao2);
tsi=1/(aodp-s4);
eta=aodp*tle->eo*tsi;
etasq=eta*eta;
eeta=tle->eo*eta;
psisq=fabs(1-etasq);
coef=qoms24*pow(tsi,4);
coef1=coef/pow(psisq,3.5);
c2=coef1*xnodp*(aodp*(1+1.5*etasq+eeta*(4+etasq))+
0.75*ck2*tsi/psisq*x3thm1*(8+3*etasq*(8+etasq)));
c1=tle->bstar*c2;
sinio=sin(tle->xincl);
a3ovk2=-xj3/ck2*pow(ae,3);
c3=coef*tsi*a3ovk2*xnodp*ae*sinio/tle->eo;
x1mth2=1-theta2;

c4=2*xnodp*coef1*aodp*betao2*(eta*(2+0.5*etasq)+tle->eo*(0.5+2*etasq)-
2*ck2*tsi/(aodp*psisq)*(-3*x3thm1*(1-2*eeta+etasq*(1.5-0.5*eeta))+
0.75*x1mth2*(2*etasq-eeta*(1+etasq))*cos(2*tle->omegao)));
c5=2*coef1*aodp*betao2*(1+2.75*(etasq+eeta)+eeta*etasq);

theta4=theta2*theta2;
temp1=3*ck2*pinvsq*xnodp;
temp2=temp1*ck2*pinvsq;
temp3=1.25*ck4*pinvsq*pinvsq*xnodp;
xmdot=xnodp+0.5*temp1*betao*x3thm1+0.0625*temp2*betao*(13-78*theta2
+137*theta4);
x1m5th=1-5*theta2;
omgdot=-0.5*temp1*x1m5th+0.0625*temp2*(7-114*theta2+395*theta4)+temp3*
(3-36*theta2+49*theta4);
xhdot1=-temp1*cosio;
xnodot=xhdot1+(0.5*temp2*(4-19*theta2)+2*temp3*(3-7*theta2))*cosio;
omgcof=tle->bstar*c3*cos(tle->omegao);
xmcof=-tothrd*coef*tle->bstar*ae/eeta;
xnodcf=3.5*betao2*xhdot1*c1;
t2cof=1.5*c1;
xlcof=0.125*a3ovk2*sinio*(3+5*cosio)/(1+cosio);
aycof=0.25*a3ovk2*sinio;
delmo=pow(1+eta*cos(tle->xmo),3);
sinmo=sin(tle->xmo);
x7thm1=7*theta2-1;

if (isFlagClear(SIMPLE_FLAG))
{
c1sq=c1*c1;
d2=4*aodp*tsi*c1sq;
temp=d2*tsi*c1/3;
d3=(17*aodp+s4)*temp;
d4=0.5*temp*aodp*tsi*(221*aodp+31*s4)*c1;
t3cof=d2+2*c1sq;
t4cof=0.25*(3*d3+c1*(12*d2+10*c1sq));
t5cof=0.2*(3*d4+12*c1*d3+6*d2*d2+15*c1sq*(2*d2+c1sq));
}
}

/* update for secular gravity and atmospheric drag. */
xmdf=tle->xmo+xmdot*tsince;
omgadf=tle->omegao+omgdot*tsince;
xnoddf=tle->xnodeo+xnodot*tsince;
omega=omgadf;
xmp=xmdf;
tsq=tsince*tsince;
xnode=xnoddf+xnodcf*tsq;
tempa=1-c1*tsince;
tempe=tle->bstar*c4*tsince;
templ=t2cof*tsq;

if (isFlagClear(SIMPLE_FLAG))
{
delomg=omgcof*tsince;
delm=xmcof*(pow(1+eta*cos(xmdf),3)-delmo);
temp=delomg+delm;
xmp=xmdf+temp;
omega=omgadf-temp;
tcube=tsq*tsince;
tfour=tsince*tcube;
tempa=tempa-d2*tsq-d3*tcube-d4*tfour;
tempe=tempe+tle->bstar*c5*(sin(xmp)-sinmo);
templ=templ+t3cof*tcube+tfour*(t4cof+tsince*t5cof);
}

a=aodp*pow(tempa,2);
e=tle->eo-tempe;
xl=xmp+omega+xnode+xnodp*templ;
beta=sqrt(1-e*e);
xn=xke/pow(a,1.5);

/* long period periodics */
axn=e*cos(omega);
temp=1/(a*beta*beta);
xll=temp*xlcof*axn;
aynl=temp*aycof;
xlt=xl+xll;
ayn=e*sin(omega)+aynl;

/* solve Kepler’s equation (numerical) */
capu=FMod2p(xlt-xnode);
temp2=capu;
i=0;

do
{
sinepw=sin(temp2);
cosepw=cos(temp2);
temp3=axn*sinepw;
temp4=ayn*cosepw;
temp5=axn*cosepw;
temp6=ayn*sinepw;
epw=(capu-temp4+temp3-temp2)/(1-temp5-temp6)+temp2;

if (fabs(epw-temp2)<= e6a)
break;

temp2=epw;

} while (i++<10);

/* short period preliminary quantities */
ecose=temp5+temp6;
esine=temp3-temp4;
elsq=axn*axn+ayn*ayn;
temp=1-elsq;
pl=a*temp;
r=a*(1-ecose);
temp1=1/r;
rdot=xke*sqrt(a)*esine*temp1;
rfdot=xke*sqrt(pl)*temp1;
temp2=a*temp1;
betal=sqrt(temp);
temp3=1/(1+betal);
cosu=temp2*(cosepw-axn+ayn*esine*temp3);
sinu=temp2*(sinepw-ayn-axn*esine*temp3);
u=AcTan(sinu,cosu);
sin2u=2*sinu*cosu;
cos2u=2*cosu*cosu-1;
temp=1/pl;
temp1=ck2*temp;
temp2=temp1*temp;

/* update for short periodics */
rk=r*(1-1.5*temp2*betal*x3thm1)+0.5*temp1*x1mth2*cos2u;
uk=u-0.25*temp2*x7thm1*sin2u;
```

```
xnodek=xnode+1.5*temp2*cosio*sin2u;
xinck=tle->xincl+1.5*temp2*cosio*sinio*cos2u;
rdotk=rdot-xn*temp1*x1mth2*sin2u;
rfdotk=rfdot+xn*temp1*(x1mth2*cos2u+1.5*x3thm1);

/* orientation vectors */
sinuk=sin(uk);
cosuk=cos(uk);
sinik=sin(xinck);
cosik=cos(xinck);
sinnok=sin(xnodek);
cosnok=cos(xnodek);
xmx=-sinnok*cosik;
xmy=cosnok*cosik;
ux=xmx*sinuk+cosnok*cosuk;
uy=xmy*sinuk+sinnok*cosuk;
uz=sinik*sinuk;
vx=xmx*cosuk-cosnok*sinuk;
vy=xmy*cosuk-sinnok*sinuk;
vz=sinik*cosuk;

/* position and velocity */
position[0]=rk*ux;
position[1]=rk*uy;
position[2]=rk*uz;

velocity[0]=rdotk*ux+rfdotk*vx;
velocity[1]=rdotk*uy+rfdotk*vy;
velocity[2]=rdotk*uz+rfdotk*vz;

/* phase in radians */
phase=xlt-xnode-omgadf+twopi;

if (phase<0.0)
phase+=twopi;

phase=FMod2p(phase);
}



/*--------------------------------------------------------------------------------------------------------------------------------*/
/******************************************** SDP4 ********************************************************************************/
/*--------------------------------------------------------------------------------------------------------------------------------*/

/* SDP4 routine to to add lunar and solar perturbation effects to deep-space */
/* orbit objects (see thesis appendix A.4.2)                                 */
void Deep(int ientry, tle_t * tle, deep_arg_t * deep_arg)
{
static double thgr, xnq, xqncl, omegaq, zmol, zmos, savtsn, ee2, e3,
xi2, xl2, xl3, xl4, xgh2, xgh3, xgh4, xh2, xh3, sse, ssi, ssg, xi3,
se2, si2, sl2, sgh2, sh2, se3, si3, sl3, sgh3, sh3, sl4, sgh4, ssl,
ssh, d3210, d3222, d4410, d4422, d5220, d5232, d5421, d5433, del1,
del2, del3, fasx2, fasx4, fasx6, xlamo, xfact, xni, atime, stepp,
stepn, step2, preep, pl, sghs, xli, d2201, d2211, sghl, sh1, pinc,
pe, shs, zsingl, zcosgl, zsinhl, zcoshl, zsinil, zcosil;

double a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, ainv2, alfdp, aqnv,
sgh, sini2, sinis, sinok, sh, si, sil, day, betdp, dalf, bfact, c,
cc, cosis, cosok, cosq, ctem, f322, zx, zy, dbet, dls, eoc, eq, f2,
f220, f221, f3, f311, f321, xnoh, f330, f441, f442, f522, f523,
f542, f543, g200, g201, g211, pgh, ph, s1, s2, s3, s4, s5, s6, s7,
se, sel, ses, xls, g300, g310, g322, g410, g422, g520, g521, g532,
g533, gam, sinq, sinzf, sis, sl, sll, sls, stem, temp, temp1, x1,
x2, x2li, x2omi, x3, x4, x5, x6, x7, x8, xl, xldot, xmao, xnddt,
xndot, xno2, xnodce, xnoi, xomi, xpidot, z1, z11, z12, z13, z2,
z21, z22, z23, z3, z31, z32, z33, ze, zf, zm, zmo, zn, zsing,
zsinh, zsini, zcosg, zcosh, zcosi, delt=0, ft=0;

switch (ientry)
{
case dpinit:  /* entrance for deep space initialisation */
thgr=ThetaG(tle->epoch,deep_arg);
eq=tle->eo;
xnq=deep_arg->xnodp;
aqnv=1/deep_arg->aodp;
xqncl=tle->xincl;
xmao=tle->xmo;
xpidot=deep_arg->omgdot+deep_arg->xnodot;
sinq=sin(tle->xnodeo);
cosq=cos(tle->xnodeo);
omegaq=tle->omegao;

/* initialize lunar solar terms */
day=deep_arg->ds50+18261.5;  /* days since 1900 Jan 0.5 */

if (day!=preep)
{
preep=day;
xnodce=4.5236020-9.2422029E-4*day;
stem=sin(xnodce);
ctem=cos(xnodce);
zcosil=0.91375164-0.03568096*ctem;
zsinil=sqrt(1-zcosil*zcosil);
zsinhl=0.089683511*stem/zsinil;
zcoshl=sqrt(1-zsinhl*zsinhl);
c=4.7199672+0.22997150*day;
gam=5.8351514+0.0019443680*day;
zmol=FMod2p(c-gam);
```

```
zx=0.39785416*stem/zsinil;
zy=zcoshl*ctem+0.91744867*zsinhl*stem;
zx=AcTan(zx,zy);
zx=gam+zx-xnodce;
zcosgl=cos(zx);
zsingl=sin(zx);
zmos=6.2565837+0.017201977*day;
zmos=FMod2p(zmos);
    }

  /* do solar terms */
  savtsn=1E20;
  zcosg=zcosgs;
  zsing=zsings;
  zcosi=zcosis;
  zsini=zsinis;
  zcosh=cosq;
  zsinh= sinq;
  cc=c1ss;
  zn=zns;
  ze=zes;
  zmo=zmos;
  xnoi=1/xnq;

  /* loop breaks when Solar terms are done a second */
  /* time, after Lunar terms are initialized        */

for (;;)
{
/* solar terms done again after Lunar terms are done */
a1=zcosg*zcosh+zsing*zcosi*zsinh;
a3=-zsing*zcosh+zcosg*zcosi*zsinh;
a7=-zcosg*zsinh+zsing*zcosi*zcosh;
a8=zsing*zsini;
a9=zsing*zsinh+zcosg*zcosi*zcosh;
a10=zcosg*zsini;
a2=deep_arg->cosio*a7+deep_arg->sinio*a8;
a4=deep_arg->cosio*a9+deep_arg->sinio*a10;
a5=-deep_arg->sinio*a7+deep_arg->cosio*a8;
a6=-deep_arg->sinio*a9+deep_arg->cosio*a10;
x1=a1*deep_arg->cosg+a2*deep_arg->sing;
x2=a3*deep_arg->cosg+a4*deep_arg->sing;
x3=-a1*deep_arg->sing+a2*deep_arg->cosg;
x4=-a3*deep_arg->sing+a4*deep_arg->cosg;
x5=a5*deep_arg->sing;
x6=a6*deep_arg->sing;
x7=a5*deep_arg->cosg;
x8=a6*deep_arg->cosg;
z31=12*x1*x1-3*x3*x3;
z32=24*x1*x2-6*x3*x4;
z33=12*x2*x2-3*x4*x4;
z1=3*(a1*a1+a2*a2)+z31*deep_arg->eosq;
z2=6*(a1*a3+a2*a4)+z32*deep_arg->eosq;
z3=3*(a3*a3+a4*a4)+z33*deep_arg->eosq;
z11=-6*a1*a5+deep_arg->eosq*(-24*x1*x7-6*x3*x5);
z12=-6*(a1*a6+a3*a5)+deep_arg->eosq*(-24*(x2*x7+x1*x8)-6*(x3*x6
+x4*x5));
z13=-6*a3*a6+deep_arg->eosq*(-24*x2*x8-6*x4*x6);
z21=6*a2*a5+deep_arg->eosq*(24*x1*x5-6*x3*x7);
z22=6*(a4*a5+a2*a6)+deep_arg->eosq*(24*(x2*x5+x1*x6)-6*(x4*x7+
x3*x8));
z23=6*a4*a6+deep_arg->eosq*(24*x2*x6-6*x4*x8);
z1=z1+z1+deep_arg->betao2*z31;
z2=z2+z2+deep_arg->betao2*z32;
z3=z3+z3+deep_arg->betao2*z33;
s3=cc*xnoi;
s2=-0.5*s3/deep_arg->betao;
s4=s3*deep_arg->betao;
s1=-15*eq*s4;
s5=x1*x3+x2*x4;
s6=x2*x3+x1*x4;
s7=x2*x4-x1*x3;
se=s1*zn*s5;
si=s2*zn*(z11+z13);
sl=-zn*s3*(z1+z3-14-6*deep_arg->eosq);
sgh=s4*zn*(z31+z33-6);
sh=-zn*s2*(z21+z23);

if (xqncl<5.2359877E-2)
sh=0;

ee2=2*s1*s6;
e3=2*s1*s7;
xi2=2*s2*z12;
xi3=2*s2*(z13-z11);
xl2=-2*s3*z2;
xl3=-2*s3*(z3-z1);
xl4=-2*s3*(-21-9*deep_arg->eosq)*ze;
xgh2=2*s4*z32;
xgh3=2*s4*(z33-z31);
xgh4=-18*s4*ze;
xh2=-2*s2*z22;
xh3=-2*s2*(z23-z21);

if (isFlagSet(LUNAR_TERMS_DONE_FLAG))
break;

/* do lunar terms */
sse=se;
```

```
ssi=si;
ssl=sl;
ssh=sh/deep_arg->sinio;
ssg=sgh-deep_arg->cosio*ssh;
se2=ee2;
si2=xi2;
sl2=xl2;
sgh2=xgh2;
sh2=xh2;
se3=e3;
si3=xi3;
sl3=xl3;
sgh3=xgh3;
sh3=xh3;
sl4=xl4;
sgh4=xgh4;
zcosg=zcosgl;
zsing=zsingl;
zcosi=zcosil;
zsini=zsinil;
zcosh=zcoshl*cosq+zsinhl*sinq;
zsinh=sinq*zcoshl-cosq*zsinhl;
zn=znl;
cc=c1l;
ze=zel;
zmo=zmol;
SetFlag(LUNAR_TERMS_DONE_FLAG);
}

sse=sse+se;
ssi=ssi+si;
ssl=ssl+sl;
ssg=ssg+sgh-deep_arg->cosio/deep_arg->sinio*sh;
ssh=ssh+sh/deep_arg->sinio;

/* geopotential resonance initialisation for 12 hour orbits */
ClearFlag(RESONANCE_FLAG);
ClearFlag(SYNCHRONOUS_FLAG);

if (!((xnq<0.0052359877) && (xnq>0.0034906585)))
{
if ((xnq<0.00826) || (xnq>0.00924))
    return;

if (eq<0.5)
    return;

SetFlag(RESONANCE_FLAG);
eoc=eq*deep_arg->eosq;
g201=-0.306-(eq-0.64)*0.440;

if (eq<=0.65)
{
  g211=3.616-13.247*eq+16.290*deep_arg->eosq;
  g310=-19.302+117.390*eq-228.419*deep_arg->eosq+156.591*eoc;
  g322=-18.9068+109.7927*eq-214.6334*deep_arg->eosq+146.5816*eoc;
  g410=-41.122+242.694*eq-471.094*deep_arg->eosq+313.953*eoc;
  g422=-146.407+841.880*eq-1629.014*deep_arg->eosq+1083.435 *eoc;
  g520=-532.114+3017.977*eq-5740*deep_arg->eosq+3708.276*eoc;
}

else
{
 g211=-72.099+331.819*eq-508.738*deep_arg->eosq+266.724*eoc;
 g310=-346.844+1582.851*eq-2415.925*deep_arg->eosq+1246.113*eoc;
 g322=-342.585+1554.908*eq-2366.899*deep_arg->eosq+1215.972*eoc;
 g410=-1052.797+4758.686*eq-7193.992*deep_arg->eosq+3651.957*eoc;
 g422=-3581.69+16178.11*eq-24462.77*deep_arg->eosq+12422.52*eoc;

if (eq<=0.715)
 g520=1464.74-4664.75*eq+3763.64*deep_arg->eosq;

else
 g520=-5149.66+29936.92*eq-54087.36*deep_arg->eosq+31324.56*eoc;
}

if (eq<0.7)
{
 g533=-919.2277+4988.61*eq-9064.77*deep_arg->eosq+5542.21*eoc;
 g521=-822.71072+4568.6173*eq-8491.4146*deep_arg->eosq+5337.524*eoc;
 g532=-853.666+4690.25*eq-8624.77*deep_arg->eosq+5341.4*eoc;
}

else
{
 g533=-37995.78+161616.52*eq-229838.2*deep_arg->eosq+109377.94*eoc;
 g521 =-51752.104+218913.95*eq-309468.16*deep_arg->eosq+146349.42*eoc;
 g532 =-40023.88+170470.89*eq-242699.48*deep_arg->eosq+115605.82*eoc;
}

sini2=deep_arg->sinio*deep_arg->sinio;
f220=0.75*(1+2*deep_arg->cosio+deep_arg->theta2);
f221=1.5*sini2;
f321=1.875*deep_arg->sinio*(1-2*deep_arg->cosio-3*deep_arg->theta2);
f322=-1.875*deep_arg->sinio*(1+2*deep_arg->cosio-3*deep_arg->theta2);
f441=35*sini2*f220;
f442=39.3750*sini2*sini2;
f522=9.84375*deep_arg->sinio*(sini2*(1-2*deep_arg->cosio-5*
deep_arg->theta2)+0.33333333*(-2+4*deep_arg->cosio+6*deep_arg->theta2));
```

```
f523=deep_arg->sinio*(4.92187512*sini2*(-2-4*deep_arg->cosio+10*
deep_arg->theta2)+6.56250012*(1+2*deep_arg->cosio-3*deep_arg->theta2));
f542=29.53125*deep_arg->sinio*(2-8*deep_arg->cosio+deep_arg->theta2*
(-12+8*deep_arg->cosio+10*deep_arg->theta2));
f543=29.53125*deep_arg->sinio*(-2-8*deep_arg->cosio+deep_arg->theta2*
(12+8*deep_arg->cosio-10*deep_arg->theta2));
xno2=xnq*xnq;
ainv2=aqnv*aqnv;
temp1=3*xno2*ainv2;
temp=temp1*root22;
d2201=temp*f220*g201;
d2211=temp*f221*g211;
temp1=temp1*aqnv;
temp=temp1*root32;
d3210=temp*f321*g310;
d3222=temp*f322*g322;
temp1=temp1*aqnv;
temp=2*temp1*root44;
d4410=temp*f441*g410;
d4422=temp*f442*g422;
temp1=temp1*aqnv;
temp=temp1*root52;
d5220=temp*f522*g520;
d5232=temp*f523*g532;
temp=2*temp1*root54;
d5421=temp*f542*g521;
d5433=temp*f543*g533;
xlamo=xmao+tle->xnodeo+tle->xnodeo-thgr-thgr;
bfact=deep_arg->xmdot+deep_arg->xnodot+deep_arg->xnodot-thdt-thdt;
bfact=bfact+ssl+ssh+ssh;
}

else
{
SetFlag(RESONANCE_FLAG);
SetFlag(SYNCHRONOUS_FLAG);

/* synchronous resonance terms initialisation */
g200=1+deep_arg->eosq*(-2.5+0.8125*deep_arg->eosq);
g310=1+2*deep_arg->eosq;
g300=1+deep_arg->eosq*(-6+6.60937*deep_arg->eosq);
f220=0.75*(1+deep_arg->cosio)*(1+deep_arg->cosio);
f311=0.9375*deep_arg->sinio*deep_arg->sinio*(1+3*deep_arg->cosio)
-0.75*(1+deep_arg->cosio);
f330=1+deep_arg->cosio;
f330=1.875*f330*f330*f330;
del1=3*xnq*xnq*aqnv*aqnv;
del2=2*del1*f220*g200*q22;
del3=3*del1*f330*g300*q33*aqnv;
del1=del1*f311*g310*q31*aqnv;
fasx2=0.13130908;
fasx4=2.8843198;
fasx6=0.37448087;
xlamo=xmao+tle->xnodeo+tle->omegao-thgr;
bfact=deep_arg->xmdot+xpidot-thdt;
bfact=bfact+ssl+ssg+ssh;
}

xfact=bfact-xnq;

/* initialize integrator */
xli=xlamo;
xni=xnq;
atime=0;
stepp=720;
stepn=-720;
step2=259200;

return;

case dpsec:  /* entrance for deep space secular effects */
deep_arg->xll=deep_arg->xll+ssl*deep_arg->t;
deep_arg->omgadf=deep_arg->omgadf+ssg*deep_arg->t;
deep_arg->xnode=deep_arg->xnode+ssh*deep_arg->t;
deep_arg->em=tle->eo+sse*deep_arg->t;
deep_arg->xinc=tle->xincl+ssi*deep_arg->t;

if (deep_arg->xinc<0)
{
deep_arg->xinc=-deep_arg->xinc;
deep_arg->xnode=deep_arg->xnode+pi;
deep_arg->omgadf=deep_arg->omgadf-pi;
}

if (isFlagClear(RESONANCE_FLAG))
      return;

do
{
  if ((atime==0)||((deep_arg->t>=0)&&(atime<0))||((deep_arg->t<0)&&(atime>=0)))
{
/* epoch restart */
if (deep_arg->t>=0)
delt=stepp;
else
delt=stepn;

atime=0;
xni=xnq;
xli=xlamo;
}

else
{
if (fabs(deep_arg->t)>=fabs(atime))
{
if (deep_arg->t>0)
delt=stepp;
else
delt=stepn;
}
}

do
{
if (fabs(deep_arg->t-atime)>=stepp)
{
SetFlag(DO_LOOP_FLAG);
ClearFlag(EPOCH_RESTART_FLAG);
}

else
{
ft=deep_arg->t-atime;
ClearFlag(DO_LOOP_FLAG);
}

if (fabs(deep_arg->t)<fabs(atime))
{
if (deep_arg->t>=0)
delt=stepn;
else
delt=stepp;

SetFlag(DO_LOOP_FLAG | EPOCH_RESTART_FLAG);
}

/* dot terms calculated */
if (isFlagSet(SYNCHRONOUS_FLAG))
{
xndot=del1*sin(xli-fasx2)+del2*sin(2*(xli-fasx4))+del3*
sin(3*(xli-fasx6));
xnddt=del1*cos(xli-fasx2)+2*del2*cos(2*(xli-fasx4))+3*del3*
cos(3*(xli-fasx6));
}

else
{
xomi=omegaq+deep_arg->omgdot*atime;
x2omi=xomi+xomi;
x2li=xli+xli;
xndot=d2201*sin(x2omi+xli-g22)+d2211*sin(xli-g22)+d3210
*sin(xomi+xli-g32)+d3222*sin(-xomi+xli-g32)+d4410*
sin(x2omi+x2li-g44)+d4422*sin(x2li-g44)+d5220*sin(xomi+xli-g52)+d5232*
sin(-xomi+xli-g52)+d5421*sin(xomi+x2li-g54)+d5433*sin(-xomi+x2li-g54);
xnddt=d2201*cos(x2omi+xli-g22)+d2211*cos(xli-g22)+d3210*cos(xomi+xli-g32)+
d3222*cos(-xomi+xli-g32)+d5220*cos(xomi+xli-g52)+d5232*cos(-xomi+xli-g52)
+2*(d4410*cos(x2omi+x2li-g44)+d4422*cos(x2li-g44)+d5421*cos(xomi+x2li-g54)
+d5433*cos(-xomi+x2li-g54));
}

xldot=xni+xfact;
xnddt=xnddt*xldot;

if (isFlagSet(DO_LOOP_FLAG))
{
xli=xli+xldot*delt+xndot*step2;
xni=xni+xndot*delt+xnddt*step2;
atime=atime+delt;
}
} while (isFlagSet(DO_LOOP_FLAG) && isFlagClear(EPOCH_RESTART_FLAG));
} while (isFlagSet(DO_LOOP_FLAG) && isFlagSet(EPOCH_RESTART_FLAG));

deep_arg->xn=xni+xndot*ft+xnddt*ft*ft*0.5;
xl=xli+xldot*ft+xndot*ft*ft*0.5;
temp=-deep_arg->xnode+thgr+deep_arg->t*thdt;

if (isFlagClear(SYNCHRONOUS_FLAG))
deep_arg->xll=xl+temp+temp;
    else
deep_arg->xll=xl-deep_arg->omgadf+temp;

return;

case dpper:  /* entrance for lunar-solar periodics */
sinis=sin(deep_arg->xinc);
cosis=cos(deep_arg->xinc);

if (fabs(savtsn-deep_arg->t)>=30)
{
savtsn=deep_arg->t;
zm=zmos+zns*deep_arg->t;
zf=zm+2*zes*sin(zm);
sinzf=sin(zf);
f2=0.5*sinzf*sinzf-0.25;
f3=-0.5*sinzf*cos(zf);
ses=se2*f2+se3*f3;
sis=si2*f2+si3*f3;
```

```
sls=sl2*f2+sl3*f3+sl4*sinzf;
sghs=sgh2*f2+sgh3*f3+sgh4*sinzf;
shs=sh2*f2+sh3*f3;
zm=zmol+znl*deep_arg->t;
zf=zm+2*zel*sin(zm);
sinzf=sin(zf);
f2=0.5*sinzf*sinzf-0.25;
f3=-0.5*sinzf*cos(zf);
sel=ee2*f2+e3*f3;
sil=xi2*f2+xi3*f3;
sll=xl2*f2+xl3*f3+xl4*sinzf;
sghl=xgh2*f2+xgh3*f3+xgh4*sinzf;
sh1=xh2*f2+xh3*f3;
pe=ses+sel;
pinc=sis+sil;
pl=sls+sll;
}

pgh=sghs+sghl;
ph=shs+sh1;
deep_arg->xinc=deep_arg->xinc+pinc;
deep_arg->em=deep_arg->em+pe;

if (xqncl>=0.2)
{
/* apply periodics directly */
ph=ph/deep_arg->sinio;
pgh=pgh-deep_arg->cosio*ph;
deep_arg->omgadf=deep_arg->omgadf+pgh;
deep_arg->xnode=deep_arg->xnode+ph;
deep_arg->xll=deep_arg->xll+pl;
}

else
{
/* apply periodics with Lyddane modification */
sinok=sin(deep_arg->xnode);
cosok=cos(deep_arg->xnode);
alfdp=sinis*sinok;
betdp=sinis*cosok;
dalf=ph*cosok+pinc*cosis*sinok;
dbet=-ph*sinok+pinc*cosis*cosok;
alfdp=alfdp+dalf;
betdp=betdp+dbet;
deep_arg->xnode=FMod2p(deep_arg->xnode);
xls=deep_arg->xll+deep_arg->omgadf+cosis*deep_arg->xnode;
dls=pl+pgh-pinc*deep_arg->xnode*sinis;
xls=xls+dls;
xnoh=deep_arg->xnode;
deep_arg->xnode=AcTan(alfdp,betdp);

/* this is a patch to Lyddane modification */
if (fabs(xnoh-deep_arg->xnode)>pi)
{
      if (deep_arg->xnode<xnoh)
  deep_arg->xnode+=twopi;
      else
  deep_arg->xnode-=twopi;
}

deep_arg->xll=deep_arg->xll+pl;
deep_arg->omgadf=xls-deep_arg->xll-cos(deep_arg->xinc)*deep_arg->xnode;
}
return;
}
}

/* SGP 4 implementation. tsince is time since epoch in minutes, tle is a     */
/* pointer to a tle_t structure with Keplerian orbital elements and position */
/* and velocity are the returning ECI satellite position and                 */
/* velocity in m and m/s.                                                    */
void SDP4(double tsince, tle_t * tle, double *position, double *velocity)
{
int i;

static double x3thm1, c1, x1mth2, c4, xnodcf, t2cof, xlcof,
aycof, x7thm1;

double a, axn, ayn, aynl, beta, betal, capu, cos2u, cosepw, cosik,
cosnok, cosu, cosuk, ecose, elsq, epw, esine, pl, theta4, rdot,
rdotk, rfdot, rfdotk, rk, sin2u, sinepw, sinik, sinnok, sinu,
sinuk, tempe, templ, tsq, u, uk, ux, uy, uz, vx, vy, vz, xinck, xl,
xlt, xmam, xmdf, xmx, xmy, xnoddf, xnodek, xll, a1, a3ovk2, ao, c2,
coef, coef1, x1m5th, xhdot1, del1, r, delo, eeta, eta, etasq,
perigee, psisq, tsi, qoms24, s4, pinvsq, temp, tempa, temp1,
temp2, temp3, temp4, temp5, temp6, bx, by, bz, cx, cy, cz;

static deep_arg_t deep_arg;

/* initialisation */

if (isFlagClear(SDP4_INITIALIZED_FLAG))
{
SetFlag(SDP4_INITIALIZED_FLAG);

/* recover original mean motion (xnodp) and   */
/* semimajor axis (aodp) from input elements. */
   a1=pow(xke/tle->xno,tothrd);
deep_arg.cosio=cos(tle->xincl);
deep_arg.theta2=deep_arg.cosio*deep_arg.cosio;
x3thm1=3*deep_arg.theta2-1;
deep_arg.eosq=tle->eo*tle->eo;
deep_arg.betao2=1-deep_arg.eosq;
deep_arg.betao=sqrt(deep_arg.betao2);
del1=1.5*ck2*x3thm1/(a1*a1*deep_arg.betao*deep_arg.betao2);
ao=a1*(1-del1*(0.5*tothrd+del1*(1+134/81*del1)));
delo=1.5*ck2*x3thm1/(ao*ao*deep_arg.betao*deep_arg.betao2);
deep_arg.xnodp=tle->xno/(1+delo);
deep_arg.aodp=ao/(1-delo);

/* for perigee below 156 km, the values */
/* of s and qoms2t are altered.         */
   s4=s;
qoms24=qoms2t;
perigee=(deep_arg.aodp*(1-tle->eo)-ae)*xkmper;

if (perigee<156.0)
{
if (perigee<=98.0)
s4=20.0;
else
s4=perigee-78.0;

qoms24=pow((120-s4)*ae/xkmper,4);
s4=s4/xkmper+ae;
}

pinvsq=1/(deep_arg.aodp*deep_arg.aodp*deep_arg.betao2*deep_arg.betao2);
deep_arg.sing=sin(tle->omegao);
deep_arg.cosg=cos(tle->omegao);
tsi=1/(deep_arg.aodp-s4);
eta=deep_arg.aodp*tle->eo*tsi;
etasq=eta*eta;
eeta=tle->eo*eta;
psisq=fabs(1-etasq);
coef=qoms24*pow(tsi,4);
coef1=coef/pow(psisq,3.5);
c2=coef1*deep_arg.xnodp*(deep_arg.aodp*(1+1.5*etasq+eeta*(4+etasq))+0.75*ck2*
tsi/psisq*x3thm1*(8+3*etasq*(8+etasq)));
c1=tle->bstar*c2;
deep_arg.sinio=sin(tle->xincl);
a3ovk2=-xj3/ck2*pow(ae,3);
x1mth2=1-deep_arg.theta2;
c4=2*deep_arg.xnodp*coef1*deep_arg.aodp*deep_arg.betao2*(eta*(2+0.5*etasq)+
tle->eo*(0.5+2*etasq)-2*ck2*tsi/(deep_arg.aodp*psisq)*(-3*x3thm1*(1-2*eeta+etasq*
(1.5-0.5*eeta))+0.75*x1mth2*(2*etasq-eeta*(1+etasq))*cos(2*tle->omegao)));
theta4=deep_arg.theta2*deep_arg.theta2;
temp1=3*ck2*pinvsq*deep_arg.xnodp;
temp2=temp1*ck2*pinvsq;
temp3=1.25*ck4*pinvsq*pinvsq*deep_arg.xnodp;
deep_arg.xmdot=deep_arg.xnodp+0.5*temp1*deep_arg.betao*x3thm1+0.0625*temp2*
deep_arg.betao*(13-78*deep_arg.theta2+137*theta4);
x1m5th=1-5*deep_arg.theta2;
deep_arg.omgdot=-0.5*temp1*x1m5th+0.0625*temp2*(7-114*deep_arg.theta2+395*theta4)+
temp3*(3-36*deep_arg.theta2+49*theta4);
xhdot1=-temp1*deep_arg.cosio;
deep_arg.xnodot=xhdot1+(0.5*temp2*(4-19*deep_arg.theta2)+2*temp3*(3-7*deep_arg.theta2))*
deep_arg.cosio;
xnodcf=3.5*deep_arg.betao2*xhdot1*c1;
t2cof=1.5*c1;
xlcof=0.125*a3ovk2*deep_arg.sinio*(3+5*deep_arg.cosio)/(1+deep_arg.cosio);
aycof=0.25*a3ovk2*deep_arg.sinio;
x7thm1=7*deep_arg.theta2-1;

/* initialise DEEP */
Deep(dpinit,tle,&deep_arg);
}

/* update for secular gravity and atmospheric drag */
xmdf=tle->xmo+deep_arg.xmdot*tsince;
deep_arg.omgadf=tle->omegao+deep_arg.omgdot*tsince;
xnoddf=tle->xnodeo+deep_arg.xnodot*tsince;
tsq=tsince*tsince;
deep_arg.xnode=xnoddf+xnodcf*tsq;
tempa=1-c1*tsince;
tempe=tle->bstar*c4*tsince;
templ=t2cof*tsq;
deep_arg.xn=deep_arg.xnodp;

/* update for deep-space secular effects */
deep_arg.xll=xmdf;
deep_arg.t=tsince;

Deep(dpsec, tle, &deep_arg);

xmdf=deep_arg.xll;
a=pow(xke/deep_arg.xn,tothrd)*tempa*tempa;
deep_arg.em=deep_arg.em-tempe;
xmam=xmdf+deep_arg.xnodp*templ;

/* update for deep-space periodic effects */
deep_arg.xll=xmam;

Deep(dpper,tle,&deep_arg);

xmam=deep_arg.xll;
xl=xmam+deep_arg.omgadf+deep_arg.xnode;
beta=sqrt(1-deep_arg.em*deep_arg.em);
```

```
deep_arg.xn=xke/pow(a,1.5);

/* long period periodics */
axn=deep_arg.em*cos(deep_arg.omgadf);
temp=1/(a*beta*beta);
xll=temp*xlcof*axn;
aynl=temp*aycof;
xlt=xl+xll;
ayn=deep_arg.em*sin(deep_arg.omgadf)+aynl;

/* solve Kepler's equation */
capu=FMod2p(xlt-deep_arg.xnode);
temp2=capu;
i=0;

do
{
sinepw=sin(temp2);
cosepw=cos(temp2);
temp3=axn*sinepw;
temp4=ayn*cosepw;
temp5=axn*cosepw;
temp6=ayn*sinepw;
epw=(capu-temp4+temp3-temp2)/(1-temp5-temp6)+temp2;

if (fabs(epw-temp2)<=e6a)
break;

temp2=epw;

} while (i++<10);

/* short period preliminary quantities */
ecose=temp5+temp6;
esine=temp3-temp4;
elsq=axn*axn+ayn*ayn;
temp=1-elsq;
pl=a*temp;
r=a*(1-ecose);
temp1=1/r;
rdot=xke*sqrt(a)*esine*temp1;
rfdot=xke*sqrt(pl)*temp1;
temp2=a*temp1;
betal=sqrt(temp);
temp3=1/(1+betal);
cosu=temp2*(cosepw-axn+ayn*esine*temp3);
sinu=temp2*(sinepw-ayn-axn*esine*temp3);
u=AcTan(sinu,cosu);
sin2u=2*sinu*cosu;
cos2u=2*cosu*cosu-1;
temp=1/pl;
temp1=ck2*temp;
temp2=temp1*temp;

/* update for short periodics */
rk=r*(1-1.5*temp2*betal*x3thm1)+0.5*temp1*x1mth2*cos2u;
uk=u-0.25*temp2*x7thm1*sin2u;
xnodek=deep_arg.xnode+1.5*temp2*deep_arg.cosio*sin2u;
xinck=deep_arg.xinc+1.5*temp2*deep_arg.cosio*deep_arg.sinio*cos2u;
rdotk=rdot-deep_arg.xn*temp1*x1mth2*sin2u;
rfdotk=rfdot+deep_arg.xn*temp1*(x1mth2*cos2u+1.5*x3thm1);

/* orientation vectors */
sinuk=sin(uk);
cosuk=cos(uk);
sinik=sin(xinck);
cosik=cos(xinck);
sinnok=sin(xnodek);
cosnok=cos(xnodek);
xmx=-sinnok*cosik;
xmy=cosnok*cosik;
ux=xmx*sinuk+cosnok*cosuk;
uy=xmy*sinuk+sinnok*cosuk;
uz=sinik*sinuk;
vx=xmx*cosuk-cosnok*sinuk;
vy=xmy*cosuk-sinnok*sinuk;
vz=sinik*cosuk;

/* position and velocity */
/* position and velocity */
position[0]=rk*ux;
position[1]=rk*uy;
position[2]=rk*uy;

velocity[0]=rdotk*ux+rfdotk*vx;
velocity[1]=rdotk*uy+rfdotk*vy;
velocity[2]=rdotk*uz+rfdotk*vz;


/*  squint angle  */
if (calc_squint)
{
bx=cos(alat)*cos(alon+deep_arg.omgadf);
by=cos(alat)*sin(alon+deep_arg.omgadf);
bz=sin(alat);
cx=bx;
cy=by*cos(xinck)-bz*sin(xinck);
cz=by*sin(xinck)+bz*cos(xinck);
ax=cx*cos(xnodek)-cy*sin(xnodek);
ay=cx*sin(xnodek)+cy*cos(xnodek);
az=cz;
}

/* phase in radians */
phase=xlt-deep_arg.xnode-deep_arg.omgadf+twopi;

if (phase<0.0)
phase+=twopi;

phase=FMod2p(phase);
}
/*--------------------------------------------------------------------------------------------------------------------------------*/
/* end SGP4/SDP4 implementation                                                                                                   */
/*--------------------------------------------------------------------------------------------------------------------------------*/
```

B.4 SatPOS.m

```
function [r,rdot,lat,lon]=SatPOS(ctime,sat)
%SATPOS calculates the footprint of a satellite at a specified time in UT
%
%   SATPOS([TIME],[SAT])  where
%      [TIME] is a datevec [YYYY MM DD hh mm ss]
%      [SAT]  array of struct with the TLE data
%
%  it retruns the position r, the velocity rdot and the latitude and
%  longitude of the subsatellite point
%
%  See also:
%     orbit.m
%
% Copyright (c) 2005-08-20, P.Daum

% default input control
if nargin<2
  msgbox('no satellite data specified','info','warn');
  r=0; rdot=0; lat=0; lon=0;  % no information
  return;
end;
if nargin==0
  msgbox('no data specified','info','warn');
  r=0; rdot=0; lat=0; lon=0;  % no information
  return;
end;
tcontrol=size(ctime);
if (tcontrol(1)~=1 || tcontrol(2)~=6)
  msgbox('time format is [YYYY MM DD hh mm ss]','info','warn');
  r=0; rdot=0; lat=0; lon=0;  % no information
  return;
end;
if (isstruct(sat)==0 || isfield(sat,'tle')==0)
  msgbox('sat must be an array of struct an contain the tle data','info','warn');
  r=0; rdot=0; lat=0; lon=0;  % no information
  return;
end;

% given time convert into Julian date
jd=date2jd(ctime);

% time since epoch in minutes
tsince=(jd-sat.tle(1)).*1440;

% orbit calculation by the MEX SGP4/SDP4 implementation
[r,rdot]=orbit(sat.epochN,sat.tle(2),sat.tle(3),sat.tle(4),sat.tle(5),...
               sat.tle(6),sat.tle(7),sat.tle(8),sat.tle(9),sat.tle(10),...
               sat.tle(11),tsince);

% scale r and rdot
r    = 6.378137e3.*r;                    % in km
rdot = (6.378137e3*1.44e3/8.64e4).*rdot; % in km/s


% latitude and longitude calculation
[lat,lon]=latlon(r,jd);

% convert to degrees
lat=lat.*180./pi;
lon=lon.*180./pi;

return

% ------------------------------------------------------------------------
% ----------------- ADDITIONAL EMBEDDED FUNCTIONS ------------------------
% ------------------------------------------------------------------------

% calculates the geodetic position of an object given its ECI position pos
% and time. Reference:  The 1992 Astronomical Almanac, page K12.
function [lat,lon]=latlon(pos, ctime)
   f=3.35281066474748e-3;              % flattening factor
   xkmper= 6.378137e3;

   theta = actan2(pos(2),pos(1));       % radians
   lon = mod(theta-thetag(ctime),2*pi); % radians
   r=sqrt(pos(1).^2+pos(2).^2);
   e2=f*(2-f);
   lat=actan2(pos(3),r);                % radians

   % initial step
   phi=lat;
   c=1/sqrt(1-e2*(sin(phi)*sin(phi)));
   lat=actan2(pos(3)+xkmper*c*e2*sin(phi),r);

   while (abs(lat-phi)>=1e-10)
      phi=lat;
      c=1/sqrt(1-e2*(sin(phi)*sin(phi)));
      lat=actan2(pos(3)+xkmper*c*e2*sin(phi),r);
  end;

  if (lat>(pi/2)) lat=lat-2*pi; end;
return

% four-quadrant arctan function
function fqd=actan2(x, y)
  if (y<0)            fqd = pi+atan(x/y);    return; end;
  if (y==0 && x>0)    fqd = pi/2;            return; end;
  if (y==0 && x<=0)   fqd = 3*pi/2;          return; end;
  if (y>0 && x>0)     fqd = atan(x/y);       return; end;
  if (y>0 && x<=0)    fqd = 2*pi+atan(x/y);  return; end;
return

% calculates the Greenwich Mean Sidereal Time
function gmst=thetag(jd)
  % seconds per day
  secday  = 8.64e4;
  % Earth rotations/siderial day
  omega_E = 1.00273790934;

  UT=(jd+0.5)-floor(jd+0.5);
  jd=jd-UT;
  TU=(jd-2451545.0)/36525;

  tmp = 24110.54841+ 8640184.812866*TU +0.093104*TU.^2-0.0000062*TU.^3;
  tmp=mod(tmp+secday*omega_E*UT,secday);

  gmst = 2*pi*tmp/secday;
return

% calculates the Julian date of a given datevec
function jd = date2jd(ctime)
  a = floor((14 - ctime(2))/12);
  y = ctime(1) + 4800 - a;
  m = ctime(2) + 12*a - 3;

  jd = ctime(3) + floor((153*m + 2)/5) ...
     + y*365 + floor(y/4) - floor(y/100) + floor(y/400) - 32045 ...
     + ( ctime(6) + 60*ctime(5) + 3600*(ctime(4) - 12) )/86400;
return
```

B.5 SatTRACK.m

```
function [azi,elv,range]=SatTRACK(lat,lon,alt,ctime,sat)
%SATTRACK calculates the azimuth, elevation and altitude of a satellite
%  over a specified ground location
%
%   SATTRACK([LAT],[LON],[ALT],[TIME],[SAT])  where
%      [LAT]  latitude of the ground station
%      [LON]  longitude of the ground station
%      [ALT]  altitude of the ground station in meters with respect to the
%             WGS elipsoid
%      [TIME] is a datevec [YYYY MM DD hh mm ss]
%      [SAT]  array of struct with the TLE data
%
%  it retruns the azimuth azi, the elevation elv and the altitude range of
%  the satellite
%
%  See also:
%     orbit.m
%
% Copyright (c) 2005-08-20, P.Daum


% default input control
if nargin==0
  msgbox('no data specified','info','warn');
  azi=0; elv=0; range=0; % no information
  return;
end;
if nargin<4
  msgbox('no satellite data specified','info','warn');
  azi=0; elv=0; range=0; % no information
  return;
end;
if nargin<3
  msgbox('no satellite data and time specified','info','warn');
  azi=0; elv=0; range=0; % no information
  return;
end;
if nargin<2
```

```
  msgbox(’ground station must be given in latitude, longitude and altitude’,...
      ’info’,’warn’);
  azi=0; elv=0; range=0; % no information
  return;
end;
tcontrol=size(ctime);
if (tcontrol(1)~=1 || tcontrol(2)~=6)
  msgbox(’time format is [YYYY MM DD hh mm ss]’,’info’,’warn’);
    azi=0; elv=0; range=0; % no information
  return;
end;
if (isstruct(sat)==0 || isfield(sat,’tle’)==0)
  msgbox(’sat must be an array of struct an contain the tle data’,’info’,’warn’);
  azi=0; elv=0; range=0; % no information
  return;
end;
if (lat<-90 || lat>=90)
  msgbox(’latitude must be between -90 and 90 degrees’,’info’,’warn’);
    azi=0; elv=0; range=0; % no information
  return;
end;
if (lon<-180 || lat>=180)
  msgbox(’longitude must be between -180 and 180 degrees’,’info’,’warn’);
    azi=0; elv=0; range=0; % no information
  return;
end;

% given latitude and longitude convert to radians
lat = lat.*(pi./180);           % north positive south negative
lon = (-1).*lon.*(pi./180);     % west positive east negative

% given altitude in km
alt = alt./1000;

% given time convert into Julian date
jd=date2jd(ctime);

% time since epoch in minutes
tsince=(jd-sat.tle(1)).*1440;

% orbit calculation by the MEX SGP4/SDP4 implementation
[r,rdot]=orbit(sat.epochN,sat.tle(2),sat.tle(3),sat.tle(4),sat.tle(5),...
               sat.tle(6),sat.tle(7),sat.tle(8),sat.tle(9),sat.tle(10),...
               sat.tle(11),tsince);

% scale r and rdot
r    = 6.378137e3.*r;                    % in km
rdot = (6.378137e3*1.44e3/8.64e4).*rdot; % in km/s

% ECI position of the ground station
[xground,yground,zground]=geo2eciobs(lat,lon,alt,jd);
ground = [xground;yground;zground];

% azimuth, elevation and range calculation
[azi,elv,range]=azielev(r,ground,lat,lon,jd);

% convert to degrees
azi=azi.*180./pi;
elv=elv.*180./pi;

return;


% ------------------------------------------------------------------------
% ----------------- ADDITIONAL EMBEDDED FUNCTIONS ------------------------
% ------------------------------------------------------------------------

% calculates the azimuth, elevation and range of a satellite from a given
% ground station in ECI coordinates
function [azi,elv,range]=azielev(r,ground,lat,lon,ctime)
  % Local Mean Sidereal Time (LMST)
  thetaobs = mod(thetag(ctime)+lon,2*pi);

  % difference between the two points in ECI coordinates
  slant = r-ground;

  % range between the points Euclidean length
  range = norm(slant);

  % faster computation
  sinlat=sin(lat);
  coslat=cos(lat);
  sintheta=sin(thetaobs);
  costheta=cos(thetaobs);

  % matrix transformation.
  % Reference: Kosch, H.J., Mathematische Ergaenzung zur Einfuehrung in die Physik
  % Binomi, pp.29-ff., 1999.
  mat(1)=sinlat.*costheta.*slant(1)+sinlat.*sintheta.*slant(2)-coslat*slant(3);
  mat(2)=(-1).*sintheta*slant(1)+costheta.*slant(2);
  mat(3)=coslat.*costheta.*slant(1)+coslat.*sintheta.*slant(2)+sinlat*slant(3);

  elv=asin(mat(3)./range);

  azi=atan(((-1).*mat(2))./mat(1));

  if (mat(1)>0) azi=azi+pi;     end;
  if (azi<0)    azi=azi+2*pi;   end;
```

```
return;

% calculates the observer position in the ECI frame
% lan and lat are the geodetic position with decimal places and the alt is
% in meter
function [x,y,z]=geo2eciobs(lat,lon,alt,ctime)
  f=3.35281066474748e-3;                    % flattening factor
  xkmper= 6.378137e3;

  thetaobs = mod(thetag(ctime)+lon,2*pi);   % Local Mean Sidereal Time (LMST)
  c=1./sqrt(1+f.*(f-2).*sin(lat).*sin(lat));
  sq=(1-f).*(1-f).*c;
  achcp=(xkmper.*c+alt).*cos(lat);

  x=achcp.*cos(thetaobs);                    % kilometers
  y=achcp.*sin(thetaobs);
  z=(xkmper*sq+alt).*sin(lat);
return;

% calculates the Greenwich Mean Sidereal Time
function gmst=thetag(jd)
  % seconds per day
  secday  = 8.64e4;
  % Earth rotations/siderial day
  omega_E = 1.00273790934;

  UT=(jd+0.5)-floor(jd+0.5);
  jd=jd-UT;
  TU=(jd-2451545.0)/36525;

  tmp = 24110.54841+ 8640184.812866*TU +0.093104*TU.^2-0.0000062*TU.^3;
  tmp=mod(tmp+secday*omega_E*UT,secday);

  gmst = 2*pi*tmp/secday;
return

% calculates the Julian date of a given datevec
function jd = date2jd(ctime)
  a = floor((14 - ctime(2))/12);
  y = ctime(1) + 4800 - a;
  m = ctime(2) + 12*a - 3;

  jd = ctime(3) + floor((153*m + 2)/5) ...
     + y*365 + floor(y/4) - floor(y/100) + floor(y/400) - 32045 ...
     + ( ctime(6) + 60*ctime(5) + 3600*(ctime(4) - 12) )/86400;
return
```

B.6 `MagPOS.m`

```
function [latN,lonN,latS,lonS]=MagPOS(ctime,sat,r0,parmod,iopt)
%MAGPOS calculates the magnetic footprint of a satellite at a specified
%  time in UT in longitude and latitude on the nothern and southern
%  hemisphere respectively
%
%   MAGPOS([TIME],[SAT],[R0],[IOPT],[PARMOD])  where
%      [TIME]    is a datevec [YYYY MM DD hh mm ss]
%      [SAT]     array of struct with the TLE data
%      [R0]      height above the surface in km
%      [PARMOD]  parameter which specifies the B-field, (1) solar pressure,
%                (2) DST in nT, (3) By in nT and (4) Bz in nT
%      [IOPT]    array of Kp values, alternatively [PARMOD] can be used in
%                this case [IOPT] become a dummy
%
%  it retruns the position geographical position of the magnetic footprint
%  in latitude and longitude (1) nothern point and (2) southern point)
%
%  See also:
%     orbit.m
%
% Copyright (c) 2005-08-20, P.Daum

if (nargin<5)
  iopt=0;
  if (nargin<4)
    % default
    parmod(1) = 3;          % solar wind pressure, nPa
    parmod(2) = -20;        % Dst, nT
    parmod(3) = 3;          % By, GSM, nT
    parmod(4) = -3;         % Bz, GSM, nT
  end;
end;
if (nargin<3)
    r0=100;                  % 100 km above the surface
end;

if (length(r0)>1)
  msgbox('R0 must be a scalar in km','info','warn');
  latN=0;lonN=0;latS=0;lonS=0; % no information
  return;
end;
tcontrol=size(ctime);
if (tcontrol(1)~=1 || tcontrol(2)~=6)
  msgbox('time format is [YYYY MM DD hh mm ss]','info','warn');
  latN=0;lonN=0;latS=0;lonS=0;   % no information
  return;
end;
if (isstruct(sat)==0 || isfield(sat,'tle')==0)
  msgbox('sat must be an array of struct an contain the tle data','info','warn');
  latN=0;lonN=0;latS=0;lonS=0; % no information
  return;
end;

% set values for the GEOPACK
if length(parmod)<10 parmod(10)=0; end;
rlim=50.;
r0=1.0+(r0./6.378137e3);

% calculates the day of the year for the given UT time
dayofyear=dayfrac(ctime);

% given time convert into Julian date
jd=date2jd(ctime);

% time since epoch in minutes
tsince=(jd-sat.tle(1)).*1440;

% orbit calculation by the MEX SGP4/SDP4 implementation
[r,rdot]=orbit(sat.epochN,sat.tle(2),sat.tle(3),sat.tle(4),sat.tle(5),...
               sat.tle(6),sat.tle(7),sat.tle(8),sat.tle(9),sat.tle(10),...
               sat.tle(11),tsince);

%---------------------------------------------------------------------------
% coordinate transformation using GEOPACK routine (ECI->ECEF->GSM)
%---------------------------------------------------------------------------
% set GEOPACK global to the given time
GEOPACK_RECALC(ctime(1),dayofyear,ctime(4),ctime(5),ctime(6));

% convert the ECI ref. frame to an ECEF ref frame
[xgeo,ygeo,zgeo] = GEOPACK_GEIGEO(r(1),r(2),r(3),1);

% convert the ECEF ref. frame to GSM coordinates
[xgsm,ygsm,zgsm] = GEOPACK_GEOGSM(xgeo,ygeo,zgeo,1);


%---------------------------------------------------------------------------
% trace fieldlines
%---------------------------------------------------------------------------

% northern footprint
[XBN,YBN,ZBN,XXBN,YYBN,ZZBN,LBN] = GEOPACK_TRACE(xgsm,ygsm,zgsm,-1,rlim,r0,...
                                          iopt,parmod,'T96','GEOPACK_IGRF_GSM');
% southern footprint
[XBS,YBS,ZBS,XXBS,YYBS,ZZBS,LBS] = GEOPACK_TRACE(xgsm,ygsm,zgsm,1,rlim,r0,...
                                          iopt,parmod,'T96','GEOPACK_IGRF_GSM');
%---------------------------------------------------------------------------
% coordinate transformation using GEOPACK routine (GSM->ECEF->LAT/LON)
%---------------------------------------------------------------------------
% convert the GSM ref. frame to ECEF coordinates
[xgeoN,ygeoN,zgeoN] = GEOPACK_GEOGSM(XBN,YBN,ZBN,-1);
[xgeoS,ygeoS,zgeoS] = GEOPACK_GEOGSM(XBS,YBS,ZBS,-1);

% convert the ECEF to lat and lon
[RN,TN,FN] = GEOPACK_SPHCAR(xgeoN,ygeoN,zgeoN,-1);
[RS,TS,FS] = GEOPACK_SPHCAR(xgeoS,ygeoS,zgeoS,-1);

% nothern footprint
latN=90.-(180./pi).*TN;
lonN=(180./pi).*FN;

% southern footprint
latS=90.-(180./pi).*TS;
lonS=(180./pi).*FS;


% ------------------------------------------------------------------------
% ----------------- ADDITIONAL EMBEDDED FUNCTIONS ------------------------
% ------------------------------------------------------------------------

% calculates the day of the year by a given ctime in a datevec format
function df=dayfrac(ctime)
 refI=date2jd([ctime(1),1,0,00,00,00]);
 refII=date2jd([ctime(1),ctime(2),ctime(3),00,00,00]);
 df=abs(refI-refII);
return;

% calculates the Julian date of a given datevec
function jd = date2jd(ctime)
  a = floor((14 - ctime(2))/12);
  y = ctime(1) + 4800 - a;
  m = ctime(2) + 12*a - 3;

  jd = ctime(3) + floor((153*m + 2)/5) ...
     + y*365 + floor(y/4) - floor(y/100) + floor(y/400) - 32045 ...
     + ( ctime(6) + 60*ctime(5) + 3600*(ctime(4) - 12) )/86400;
return
```

B.7 Validation

In order to validate the output of the function contained in the toolbox, the GODDARD Space Flight Center SSC Locator was used. The `NOAA 17` satellite was simulated at the 12/08/2005 in the time between 07:25:00-08:05:00. The results are shown in table B.1 and match the results calculated with the functions `SatPOS.m`, `SatTRACK` and `MagPOS`.

Time	GEI			GEO		GSM			NB		SB	
hh:mm	x	y	z	lat	lon	x	y	z	lat	lon	lat	lon
07:25	0.33	-0.87	0.64	34.75	218.80	-0.60	0.57	0.77	38.55	220.02	-39.07	200.23
07:26	0.30	-0.83	0.70	38.24	217.72	-0.55	0.56	0.81	41.66	218.93	-41.32	197.07
07:27	0.28	-0.80	0.75	41.73	216.56	-0.49	0.56	0.85	44.81	217.76	-43.53	193.73
07:28	0.25	-0.76	0.80	45.20	215.29	-0.43	0.55	0.89	47.97	216.49	-45.71	190.21
07:29	0.22	-0.71	0.85	48.67	213.89	-0.37	0.54	0.92	51.15	215.09	-47.83	186.48
07:30	0.19	-0.67	0.89	52.13	212.32	-0.31	0.52	0.95	54.33	213.51	-49.90	182.52
07:31	0.16	-0.62	0.93	55.56	210.53	-0.25	0.51	0.98	57.52	211.71	-51.88	178.32
07:32	0.12	-0.57	0.97	58.98	208.44	-0.18	0.49	1.00	60.70	209.61	-53.77	173.87
07:33	0.09	-0.52	1.00	62.36	205.95	-0.12	0.47	1.02	63.87	207.09	-55.56	169.19
07:34	0.06	-0.46	1.03	65.70	202.90	-0.05	0.45	1.03	67.00	203.98	-57.25	164.53
07:35	0.03	-0.40	1.05	68.97	199.02	0.01	0.43	1.04	70.09	200.03	-58.72	159.66
07:36	-0.01	-0.35	1.07	72.14	193.90	0.08	0.40	1.05	73.08	194.79	-59.77	153.84
07:37	-0.04	-0.29	1.09	75.14	186.84	0.14	0.38	1.05	75.91	187.54	-60.35	146.48
07:38	-0.07	-0.23	1.10	77.84	176.63	0.21	0.35	1.05	78.45	177.08	-60.31	133.67
07:39	-0.11	-0.16	1.11	80.00	161.59	0.27	0.32	1.05	80.47	161.74	NaN	NaN
07:40	-0.14	-0.10	1.11	81.22	140.71	0.33	0.29	1.04	81.56	140.73	NaN	NaN
07:41	-0.17	-0.04	1.11	81.09	117.19	0.39	0.26	1.02	81.38	117.49	NaN	NaN
07:42	-0.20	0.02	1.11	79.67	97.29	0.45	0.23	1.01	80.00	97.98	-59.54	110.48
07:43	-0.23	0.08	1.10	77.38	83.25	0.51	0.20	0.99	77.82	84.18	-58.35	104.86
07:44	-0.26	0.15	1.09	74.61	73.73	0.57	0.16	0.96	75.21	74.73	-56.93	98.71
07:45	-0.29	0.21	1.07	71.57	67.08	0.62	0.13	0.93	72.36	68.08	-55.25	92.77
07:46	-0.32	0.27	1.05	68.38	62.23	0.67	0.09	0.90	69.38	63.17	-53.31	87.20
07:47	-0.34	0.33	1.02	65.09	58.52	0.72	0.06	0.87	66.31	59.39	-51.10	82.02
07:48	-0.37	0.39	0.99	61.74	55.59	0.77	0.02	0.83	63.19	56.37	-48.66	77.20
07:49	-0.39	0.44	0.96	58.35	53.18	0.81	-0.01	0.79	60.05	53.88	-46.02	72.69
07:50	-0.41	0.50	0.92	54.93	51.15	0.85	-0.05	0.74	56.90	51.78	-43.20	68.45
07:51	-0.43	0.55	0.88	51.49	49.41	0.88	-0.08	0.69	53.75	49.96	-40.23	64.47
07:52	-0.45	0.60	0.84	48.03	47.87	0.92	-0.12	0.64	50.61	48.36	-37.13	60.70
07:53	-0.47	0.65	0.79	44.55	46.50	0.95	-0.15	0.59	47.50	46.94	-33.93	57.15
07:54	-0.48	0.70	0.74	41.07	45.25	0.97	-0.19	0.54	44.44	45.65	-30.65	53.83
07:55	-0.50	0.74	0.69	37.58	44.10	1.00	-0.22	0.48	41.43	44.47	-27.35	50.77
07:56	-0.51	0.78	0.63	34.08	43.03	1.01	-0.26	0.42	38.51	43.38	-24.06	48.00
07:57	-0.52	0.82	0.57	30.58	42.03	1.03	-0.29	0.36	35.71	42.36	-20.87	45.58
07:58	-0.52	0.86	0.51	27.07	41.08	1.04	-0.32	0.30	33.06	41.41	-17.83	43.50
07:59	-0.53	0.89	0.45	23.56	40.18	1.05	-0.35	0.24	30.62	40.50	-15.03	41.76
08:00	-0.53	0.92	0.39	20.05	39.31	1.05	-0.38	0.17	28.48	39.64	-12.56	40.32
08:01	-0.53	0.94	0.32	16.53	38.46	1.05	-0.40	0.11	26.74	38.82	-10.53	39.11
08:02	-0.53	0.96	0.25	13.02	37.64	1.04	-0.43	0.04	25.53	38.02	-9.06	38.08
08:03	-0.53	0.98	0.19	9.50	36.83	1.03	-0.45	-0.02	24.95	37.23	-8.28	37.18
08:04	-0.52	0.99	0.12	5.98	36.03	1.02	-0.48	-0.09	25.08	36.44	-8.25	36.36
08:05	-0.52	1.00	0.05	2.46	35.24	1.00	-0.50	-0.15	25.91	35.63	-8.95	35.59

Table B.1: GODDARD SSC Locator Values

C GEOPACK

The original `GEOPACK` library is in FORTRAN and the translation to MATLAB® is done by Paul O'Brien, the changes made for **VIS SAT** are not additionally marked. Like GEOPACK is a free available code and can be changed, these code is subject to the same condition and may be modified in future studies. Here only the main header file is shown, the complete `GEOPACK` can be found on the attached CD-Rom and is marked with `GEOPACK_<func>.m`.

```
%  #########################################################################################
%  #                                                                                       #
%  #                                GEOPACK-2003                                           #
%  #                         (MAIN SET OF FORTRAN CODES)                                   #
%  #                                                                                       #
%  #          translated by PAUL O’BRIEN modified for VIS SAT by PATRICK DAUM              #
%  #########################################################################################
%
% This collection of subroutines is a result of several upgrades of the original package
% written by N. A. Tsyganenko in 1979. This version is dated April 9, 2003.
```

```
%
% This package represents an in-depth revision of the previous version, with significant
% changes in the format of calling statements. Users should familiarize themselves with
% the new formats and rules, and accordingly adjust their source codes, as specified
% below.
%
% The following changes were made to the previous release of GEOPACK (Jan 5, 2001).
%
% (1) Subroutine IGRF, calculating the Earth's main field:
%    (a) Two versions of this subroutine are provided here. In the first one (IGRF_GSM)
%        both input (position) and output (field components) are in the Geocentric Solar-
%        Magnetospheric Cartesian coordinates, while the second one (IGRF_GEO) uses sphe-
%        rical geographical (geocentric) coordinates, as in the older releases.
%
%    (b) updating of all expansion coefficients is now made separately in the s/r RECALC,
%        which also takes into account the secular change of the coefficients within
%        a given year (at the Earth's surface, the rate of the change can reach 7 nT/month).
%
%    (c) the optimal length of spherical harmonic expansions is now automatically set
%        inside the code, based on the radial distance, so that the deviation from the
%        full-length approximation does not exceed 0.01 nT. (In the previous versions,
%        the upper limit NM of the order of harmonics had to be specified by users),
%
% (2) Subroutine DIP, calculating the Earth's field in the dipole approximation:
%    (a) no longer accepts the tilt angle via the list of formal parameters. Instead,
%        the sine SPS and cosine CPS of that angle are now forwarded into DIP via the
%        first common block /GEOPACK1/.  Accordingly, there are two options: (i) to
%        calculate SPS and CPS by calling RECALC before calling DIP, or (ii) to specify
%        them explicitly. In the last case, SPS and CPS should be specified AFTER the
%        invocation of RECALC (otherwise they would be overridden by those returned by
%        RECALC).
%
%    (b) the Earth's dipole moment is now calculated by RECALC, based on the table of
%        the IGRF coefficients and their secular variation rates, for a given year and
%        the day of the year, and the obtained value of the moment is forwarded into DIP
%        via the second common block /GEOPACK2/. (In the previous versions, only a single
%        fixed value was provided for the geodipole moment, corresponding to the most
%        recent epoch).
%
% (3) Subroutine RECALC now consolidates in one module all calculations needed to
%     initialize and update the values of coefficients and quantities that vary in
%     time, either due to secular changes of the main geomagnetic field or as a result
%     of Earth's diurnal rotation and orbital motion around Sun. That allowed us to
%     simplify the codes and make them more compiler-independent.
%
% (4) Subroutine GEOMAG is now identical in its structure to other coordinate trans-
%     formation subroutines. It no longer invokes RECALC from within GEOMAG, but uses
%     precalculated values of the rotation matrix elements, obtained by a separate
%     external invocation of RECALC. This eliminates possible interference of the
%     two subroutines in the old version of the package.
%
% (5) Subroutine TRACE (and the subsidiary modules STEP and RHAND):
%    (a) no longer needs to specify the highest order of spherical harmonics in the
%        main geomagnetic field expansion - it is now calculated automatically inside the
%        IGRF_GSM (or IGRF_GEO) subroutine.
%    (b) the internal field model can now be explicitly chosen by specifying the para-
%          meter INNAME (either IGRF_GSM or DIP).
%
% (6) A new subroutine BCARSP was added, providing a conversion of Cartesian field
%     components into spherical ones (operation, inverse to that performed by the sub-
%     routine  BSPCAR).
%
% (7) Two new subroutines were added, SHUETAL_MGNP and T96_MGNP, providing the position
%     of the magnetopause, according to the model of Shue et al. [1998] and the one
%     used in the T96 magnetospheric magnetic field model.
%
% REFERENCES:
% [1] Tsyganenko, N.A., Usmanov, A.V., Determination of the Magnetospheric Current
%     System Parameters and Development of Experimental Geomagnetic Field
%     Models Based on Data from IMP and HEOS Satellites, Planet. Space Sci.,
%     Vol. 30, pp. 985-998, 1982.
%
% [2] Tsyganenko, N.A., Usmanov, A.V., Papitashvili, V.O., Papitashvili, N.E.,
%     Popov, V.A., Software for Computations of Geomagnetic Field and Related
%     Coordinate Systems, Soviet Geophysical Committee, Special Report, 58 pp.,
%     Moscow, 1987.
%
% [3] Tsyganenko, N.A., Global Quantitative Models of the Geomagnetic Field in
%     the Cislunar Magnetosphere for Different Disturbance Levels, Planet. Space
%     Sci., Vol. 35, pp. 1347-1358, 1987.
%
% [4] Tsyganenko, N.A., A Magnetospheric Magnetic Field Model with a Warped
%     Tail Current Sheet, Planet. Space Sci., Vol. 37, pp. 5-20, 1989.
%
% [5] Tsyganenko, N.A., Modeling the Earth's Magnetospheric Magnetic Field Confined
%     within a Realistic Magnetopause, J.Geophys.Res., Vol. 100, pp. 5599-
%     5612, 1995.
% [6] Tsyganenko, N.A., Stern, D.P., Modeling the Global Magnetic Field of the
%     Large-Scale Birkeland Current Systems, J. Geophys.Res., Vol. 101, pp. 27187-
%     27198, 1996.
%
% [7] Tsyganenko, N.A., Magnetic Field Model, FORTRAN Routines,
%     Geopack 2005, NASA GSFC, 2005, available under:
%     http://nssdcftp.gsfc.nasa.gov/models/magnetospheric/tsyganenko/ and
%     http://nssdc.gsfc.nasa.gov/space/model/magnetos/data-based/modeling.html
```

Acknowledgment

This thesis marks the end of my master studies. I would like to take this opportunity to thank the people who have helped me during the last years.

I would like to say thanks to all staff members from the Department of Communication Systems for all your support and the warm welcome. I would like to thank Dr. Jim A. Wild for providing the interesting topic and for his supervision during my stay, I also would like to say thank you to Dr. Andrew J. Kavanagh. A special thank you to Prof. Farideh Honary for her supervision and guidance during the Master Course in "Satellite Communication and Space Environment" and for her help for my next academic orientation in the field of physics of the Earth's Space Environment. Also I would like to thanks Prof. May-Britt Kallenrode and Dr. Bernd Heber who have inspired me to orientate my academic career in the field of space physics.

Finally I want to thank my parents and my whole family. It is you who made my studies possible. Thank you for all your help, support and encouragement you gave me.

Danksagungen

Diese Master Arbeit entstand am "Department of Communication Systems" der Universität Lancaster (Großbritannien).

Mein Dank gilt den Mitarbeitern des Departments für die Unterstützung sowie Motivation und Dr. Jim A. Wild für die ausserordentlich interessante Themenstellung und die Begleitung bei dieser Arbeit sowie Dr. Andrew J. Kavanagh. Mein besonderer Dank gilt Prof. Farideh Honary für die Leitung und interessante Zusammenstellung des Master Studienprogrammes in "Satellite Communication and Space Environment" und ihrer Hilfe bei meiner weiteren wissenschaftlichen Orientierung im Feld der Physik des erdnahen Weltraumes. Auch gilt mein Dank Prof. May-Britt Kallenrode und Dr. Bernd Heber die mich mit Ihrer Forschungsarbeit und Ihren Vorlesungen inspiriert haben, mich in diesem Bereich der Physik zu engagieren.

Ein ganz besonderer Dank gilt meinen Eltern und meiner Familie, die mir das Studium der Physik und ein Auslandsstudium erst ermöglicht haben und die mir immer finanziell und mental zu Seite gestanden haben.

Thank you
Danke
gracias
dankuwel
Ευχαριστώ
Спасибо

Department of
Communication Systems

Name: Patrick Daum

I declare that the work presented in this document is my own except where stated otherwise. The work has been carried out in the Department of Communication Systems at Lancaster University for the MSc in Satellite Communications and Space Environment.

Lancaster (U.K.), September 7, 2005

Patrick Daum

Satellite Communication Connects People

*"As far as the laws of mathematics refer to reality, they are not certain; and as far as they are certain, they do not refer to reality." (Albert Einstein *1879 - †1955)*

This shall be the closing words for my thesis. The described equations and processes are described with analytical formulas but the numeric is perhaps the best way to reproduce the reality with its included inaccuracy.