Seconda Università degli Studi di Napoli

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Aerospaziale

TESI di LAUREA

NAIF and SPICE library: an application in scenario of Post-EPS mission

Relatore
Candidato
Ch.mo Prof. Ing. **Marco D'Errico** **Salvatore Tuosto**
Dipartimento di Ingegneria Aerospaziale e Meccanica Matr. 831/247

Correlatore
Dott.ssa **M. Rosaria Santovito**
Consorzio di Ricerca su Sistemi di Telesensori Avanzati
Co.Ri.S.T.A. - Napoli

Anno Accademico 2007-2008

Per aspera sic itur ad astra.
(Seneca)

➢ TABLE OF CONTENTS

➢ TABLE OF IMAGES

➢ INTRODUCTION

This study has been developed in Naples at Co.Ri.S.T.A. (Consortium of Research on Advanced Remote Sensing System), in scenario of Post-EPS (EUMETSAT Polar System) mission. Post-EPS satellites, developed by ESA (European Space Agency) on behalf of EUMETSAT (European Organisation for the Exploitation of Meteorological Satellites) provide more precise details about atmospheric temperature and humidity profiles, fundamental for weather forecasting and climate monitoring. This programme has brought a new era in the observations of Earth's weather, climate and environment, and it will significantly improve operational meteorology, in particular Numerical Weather Prediction (NWP), able to compute forecasts ranging from a few hours up to 10 days ahead. Researchers of Co.Ri.S.T.A. are members of the science team devoted to study performances of the microwave radiometer for temperature measurements.

The purpose of this work is to analyze Post-EPS radiometer coverage and Sun and Moon illumination angles respect to the radiometer, fundamental for mission success. For this aim an innovative method has been studied and developed, never used before in this kind of mission. The whole work has been developed using SPICE library, provided by the NAIF node of NASA Planetary Data System (PDS). SPICE is a collection of data, tools, routines, software, that allows scientists and engineers to share data analysis and scientific results come from past mission to improve future mission analysis. Studying NAIF database, a good knowledge of SPICE system has been acquired, earning familiarity with this method. Using MATLAB as interface, we have developed a software, creating apposite tools and data files (called *kernels*) for this detailed mission analysis.

The thesis is organized in three chapters:

Chapter 1 introduces SPICE system and key concepts, dealing with kernels' functions, SPICE software and routines, focusing on MATLAB interface.

Chapter 2 presents Post-EPS mission and the application of the SPICE system in mission scenario, illustrating kernels structure and their generation in order to analyze radiometer geometry.

Chapter 3 shows results of illumination angles analysis and radiometer coverage.

➢ CHAPTER I: Introduction to NAIF library and fundamentals concepts

1. Planetary Data System, NAIF and SPICE

SPICE is an information system built to assist scientists and engineers in planning and interpret scientific observations, modeling, planning and executing activities needed to conduct planetary exploration missions. SPICE system includes a large range of software, mostly in the form of subroutines to incorporate in application programs in order to read SPICE data files and to compute derived observation geometry, such as altitude, latitude, longitude, and illumination angles. These software and routines are *black boxes*, the algorithm existing behind routines is not visible and they are for NAIF staff use only. SPICE is a collection of data, tools, routines, software, that allows scientists and engineers not only to make an accurate and precise mission analysis, but also to share data analysis and scientific results come from past mission to improve future mission analysis. The use of SPICE extends from mission concept development through the post-mission data analysis phase. In the following, we describe in detail SPICE system structure.

The Planetary Data System (PDS) archives and distributes scientific data from NASA planetary missions, astronomical observations, and laboratory space measurements. The PDS is organized as a federation of 8 *Nodes* and several *Subnodes* (see Fig.1.1):

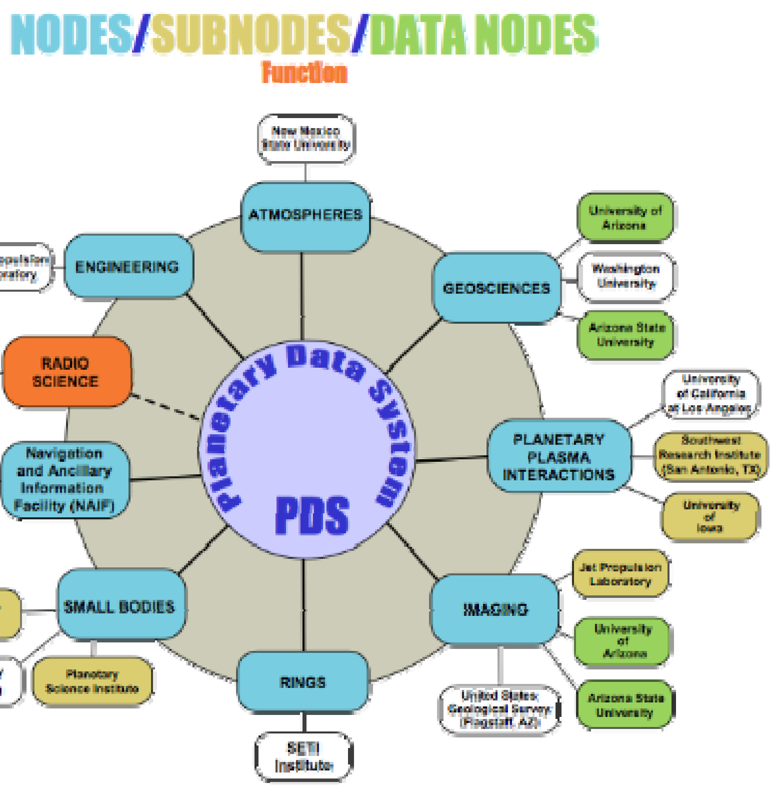

NODES/SUBNODES/DATA NODES
Function
New Mexico State University
ATMOSPHERES
Jet Propulsion Laboratory
ENGINEERING
University of Arizona
Washington University
Arizona State University
GEOSCIENCES
Stanford University
RADIO SCIENCE
Planetary Data System
PDS
University of California at Los Angeles
Southwest Research Institute (San Antonio, TX)
University of Iowa
PLANETARY PLASMA INTERACTIONS
Jet Propulsion Laboratory
Navigation and Ancillary Information Facility (NAIF)
University of Hawaii
SMALL BODIES
University of Maryland
Planetary Science Institute
RINGS
SETI Institute
Jet Propulsion Laboratory
University of Arizona
IMAGING
United States Geological Survey (Flagstaff, AZ)
Arizona State University

Fig. 1.1: Planetary Data System block diagram. [9]

- The Atmosphere Node is responsible for the acquisition, preservation, and distribution of all atmospheric data from all planetary missions (excluding Earth observations);
- the Geosciences Node deals with data that are relevant to the geosciences disciplines, the study of surfaces an interiors of terrestrial planetary bodies. Its primary goal is to ensure that the geosciences data sets coming from each planetary missions are properly documented and archived;
- the Planetary Plasma Interactions Node is responsible for the acquisition, preservation, and distribution of field and particle data from all planetary missions;

- the Imaging Node maintains and distributes the archives of planetary image data acquired from NASA's missions. Its primary goal is to enable the science community to perform image processing and analysis of the data;
- the Rings Node deals with archiving and distributing scientific data sets relevant to planetary ring systems. Most of this data sets are from Voyager missions, Hubble telescope and other Earth-based telescopes;
- the Small Bodies Node provides data from comets, asteroids an interplanetary dust;
- the Navigation and Ancillary Information Facilities (NAIF) Node is responsible for design and implementation of the SPICE concept, such as archiving, distributing and accessing observation geometry and related ancillary data used in mission design, mission evaluation, observation planning and science data analysis;
- the Engineering Node provides systems engineering support to the entire PDS, dealing with global aspects such as standards (data, software, documentation), technology investigations, catalogue development an implementation.

NAIF serves as the "ancillary data node", archiving and distributing the **SPICE kernel files** produced by several missions. NAIF also distributes generic ephemeris data for planets, satellites, comets and asteroids. An **ephemeris** is a table of values that gives the positions of astronomical objects in the sky at a given time or for intervals of time.

Ancillary data helps scientists and engineers to determine where the spacecraft is located, how its instruments are oriented, what is the location, size, shape and orientation of the target, what events are occurring on the spacecraft (or on the ground) that might affect interpretation of observations and performances of the payloads as shown in figure 1.2.

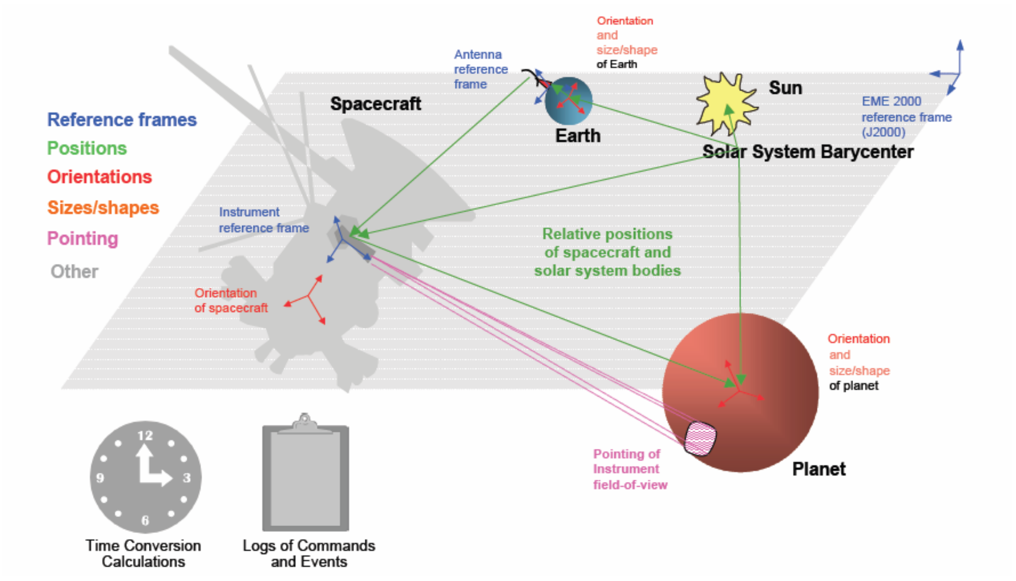

Reference frames
Positions
Orientations
Sizes/shapes
Pointing
Other
Spacecraft
Antenna reference frame
Orientation and size/shape of Earth
Earth
Sun
EME 2000 reference frame (J2000)
Solar System Barycenter
Instrument reference frame
Relative positions of spacecraft and solar system bodies
Orientation of spacecraft
Orientation and size/shape of planet
Pointing of Instrument field-of-view
Planet
Time Conversion Calculations
Logs of Commands and Events

Figure 1.2: Ancillary Data Information. [11]

Ancillary data are collected from the spacecraft, from the mission control centre, from spacecraft and instrument builders, and from the scientists. **SPICE** is used to organize and package all these data in file types –called “kernels”.

2. SPICE System

SPICE stands for “Spacecraft Planets Instrument C- Matrix Events”. The acronym is quite intuitive, except for “C” that stands for “camera”, referring to the camera installed on most of spacecrafts.
The principal SPICE system components are two:

- data files, often called **Kernels;**
- software, often known as **Spice toolkit.**

Kernels

SPICE kernels are composed of navigation and other ancillary informations that has been structured and formatted for easy access and correct use by the planetary science and engineering communities. So Kernels are files of "low level" ancillary informations that can be used, linked with other files and using the SPICE toolkit, to find out "high level" geometrical informations, as latitude, longitude, field of view and other similar data.
In the following, SPICE acronym and kernel file contents are summarized:

S- Spacecraft ephemeris, given as a function of time (SPK);

P- Planet, satellite, comet, or asteroid ephemeredes, or more generally, location of any target body, given as a function of time (PCK);

I- Instrument description kernel, containing descriptive data peculiar to a particular scientific instrument, such as field-of-view size, shape and orientation parameters (IK);

C- Pointing kernel, containing a transformation, traditionally called the C-matrix, which provides time-tagged pointing (orientation) angles for a spacecraft structure upon which science instruments are mounted. May also include angular rate data (CK);

E- Events kernel, summarizing mission activities - both planned and unanticipated (EK).The Events kernel idea has not taken hold. After Cassini it may disappear.

There are also other important components of the SPICE, even if not contained in the SPICE acronym:

- "Frame kernel" contains specifications for the assortment of reference frames that are typically used by flight projects. This file also includes mounting alignment information for instruments, antennas and perhaps other structures of interest (FK);
- "Spacecraft Clock Kernel" (SCLK),contains coefficients used for time conversion from Spacecraft Clock (SCLK) to Ephemeris Time (ET);
- "Leap seconds kernel" (LSK) contains a second tabulation used for time conversion from Coordinated Universal Time (UTC) to Ephemeris Time (ET).

SPICE kernels can be downloaded from ftp://naif.jpl.nasa.gov/pub/naif/generic_kernels.
Common usage for SPICE kernel file name extensions is:
t* text format (e.g. pck00008.tpc)
b* binary format (e.g. de421.bsp)
x* transfer format (e.g. de421.xsp)

SPICE toolkit
The SPICE system includes the SPICE Toolkit, a collection of software. The principal component of this toolkit is a library of subroutines needed to read the kernel files and to calculate

observation geometry parameters of interest. Users can integrate these SPICE toolkit subroutines into their own application such as ANSI FORTRAN 77, C, IDL and MATLAB.

Generic SPICE Toolkits have an associated version number. As of this writing (February 2009) the available version is N0062, released on march 2008. The SPICE Toolkit can be downloaded from ftp://naif.jpl.nasa.gov/pub/naif/toolkit. In this transfer folder the toolkit is available for different platforms (PC, MAC,…) , operating systems (WINDOWS, LINUX, UNIX,…) and compilers.

For the purpose of this study we analyse the SPICE Toolkit by means its interface MICE with the compiler MATLAB 7.5 (it has to be noted that NAIF built and tested MICE using MATLAB version 7.4) on a PC platform for WINDOWS.

NAIF distributes MICE as a complete, standalone package. The package includes:

- the CSPICE source files;
- the MICE interface source code;
- platform specific build scripts for MICE and CSPICE;
- an HTML based help system for both MICE and CSPICE;
- the MICE MEX shared library and the M wrapper files.

The system is ready for use after installation of the library.

The toolkit directory (directory structure for different interface is almost identical) consists of:

- **data**: cookbook example kernel (used only for training);
- **doc**: text documents and HTML documentation. A toolkit User's Guide, where everything about executable and SPICE software is explained. The extensions of these files can be *.ug (as User's

Guide) and *.req (as "Required Reading" reference documents): they can be opened with a common text editor;

- **include**: header files;
- **lib**: toolkit libraries;
- **src**: source code directories for executables and libraries;
- **exe**: utility programs. They allow to make several operations on kernel files, such as taking out comments, converting a binary format in a "transfer" one (to transfer files on computers that use different binary files storage).

  These programs are:
- **brief.exe** : command line program that displays a contents and time coverage summary for SPK or binary PCK files;
- **ckbrief.exe**: command line program that summarizes the pointing coverage for CK files;
- **commnt.exe**: command line program that reads, adds, extracts or deletes comments from SPICE binary kernel files;
- **chronos.exe**: command line program that converts between several time systems and time formats;
- **inspect.exe**: interactive program that examines the contents of an events component (ESQ) of an E-kernel;
- **mkspk.exe**: program that creates an SPK file from a text file containing trajectory informations;
- **msopck.exe**: command line program that converts attitude data provided in a text file as UTC, SCLK or ET-tagged quaternions, Euler angles or matrices, optionally accompanied by angular velocities;

- **simple.exe**: program that calculates the angular separation of two target bodies as seen from an observing body;
- **spacit.exe**: program that converts kernel in transfer format to binary format, converts binary kernels to transfer format and summarizes the contents of binary kernels;
- **spkdiff.exe**: program that computes differences between geometric states obtained from two SPK files and either displays these differences or shows statistics about them;
- **spkmerge.exe**: program that subsets or merges SPK files into a single one;
- **states.exe**: program that demonstrates the use of SPK files and subroutines by computing the state of a target body as seen from an observing body at a number of epochs within a given time interval;
- **subpt.exe**: program that demonstrates the use of CSPICE in computing the apparent sub-observer point on a target body;
- **tictoc.exe**: program that demonstrates the use of CSPICE time conversion utility routines string➔ET and ET➔UTC;
- **tobin.exe**: command line program that converts transfer format SPK, CK and EK files to binary format;
- **toxfr.exe**: command line program that converts binary format SPK, CK, EK files to transfer format;

3. Key concepts and definitions

In addition to the description of SPICE library and its utility programs, it seems important also to clarify the definitions of fundamentals concepts, like time, reference frames and aberration corrections in order to avoid misunderstanding in the following.

Time

Time is the fundamental dimension in almost every branch of science. The basis for scientific time is a continuous count of second based on two hundred atomic clocks in over fifty national laboratories, known as the International Atomic Time (TAI). Due to the averaging, it's far more stable than any clock would be alone. The **Atomic Time** counts simply atomic seconds past the astronomically determined instant of midnight (00:00:00) of 1st January 1958 at Royal Observatory of Greenwich. The standard that gives a name to each second of TAI is known as **Universal Coordinated Time** (UTC). This standard is the basis of modern civil time. UTC dates are represented as strings, such as

"26 JULY 1986 1:30:07.162 (UTC)"

As shown in figure 1.3, the **Universal Time** (UT1) is a timescale based on Earth rotation by observing celestial bodies crossing the meridian every day. Astronomers have preferred observing meridian crossing of stars over observations of the Sun, because these are more accurate.

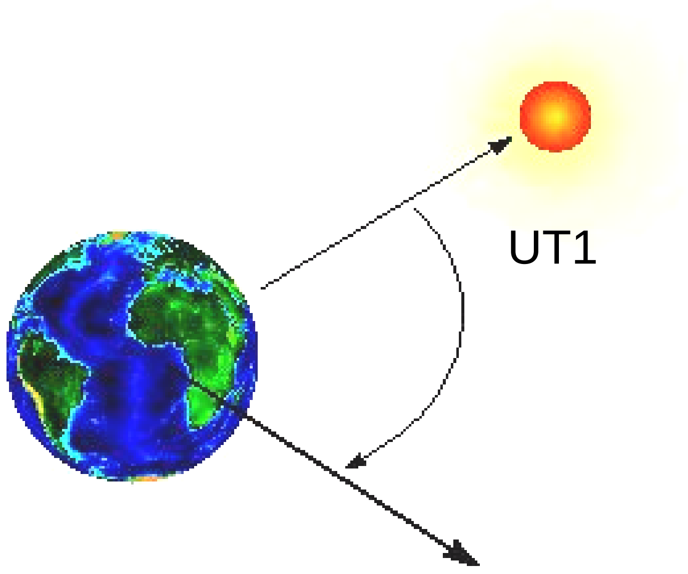


Figure 1.3: Definition of UT1. [11]

Ideally, UTC noon and astronomical noon at Greenwich (UT1) should occur simultaneously (fig. 1.4) since the Earth rotation is not uniform.

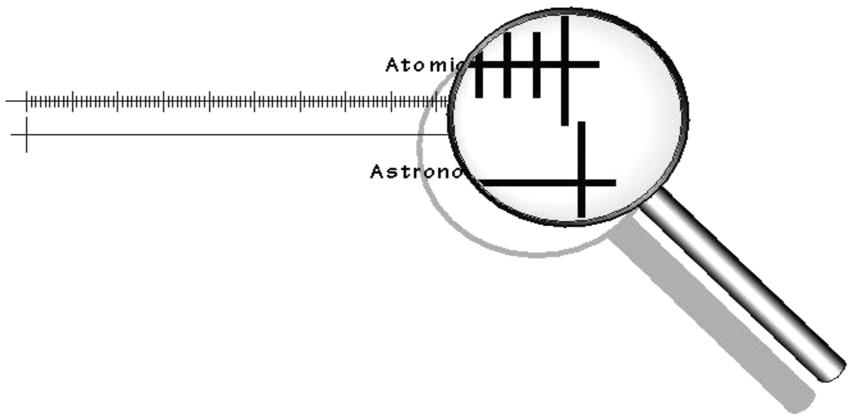


Figure 1.4: Difference between UTC and UT1 due to Earth rotation [11]

When the difference between UTC and UT1 becomes greater than 0.9 atomic seconds, a “leap second” is added (or removed). Leap seconds are normally added to the end of a designed UTC day, either June or December.

The sequence will be:

```
… DECEMBER 31 23:59:57
… DECEMBER 31 23:59:58
… DECEMBER 31 23:59:59
… DECEMBER 31 23:59:60
… JANUARY    1  00:00:00
```

Rather then

```
… DECEMBER 31 23:59:57
… DECEMBER 31 23:59:58
… DECEMBER 31 23:59:59
… JANUARY    1  00:00:00
```

Leap seconds are very important in using SPICE. For a temporal conversion, routines always require a LSK, a text file containing the leap seconds updated list.

A sample of an LSK is:

```
\begindata

DELTET/DELTA_T_A       =   32.184
DELTET/K               =    1.657D-3
DELTET/EB              =    1.671D-2
DELTET/M               = (  6.239996D0
1.99096871D-7 )

DELTET/DELTA_AT        = ( 10,   @1972-JAN-1
11,   @1972-JUL-1
12,   @1973-JAN-1
13,   @1974-JAN-1
14,   @1975-JAN-1
15,   @1976-JAN-1
16,   @1977-JAN-1
17,   @1978-JAN-1
18,   @1979-JAN-1
19,   @1980-JAN-1
20,   @1981-JUL-1
21,   @1982-JUL-1
22,   @1983-JUL-1
23,   @1985-JUL-1
24,   @1988-JAN-1
25,   @1990-JAN-1
26,   @1991-JAN-1
27,   @1992-JUL-1
28,   @1993-JUL-1
29,   @1994-JUL-1
30,   @1996-JAN-1
31,   @1997-JUL-1
32,   @1999-JAN-1
33,   @2006-JAN-1
34,   @2009-JAN-1 )

\begintext
```

Ephemeris time (ET) is an uniform timescale used in ephemerides of celestial bodies. Two kinds of ephemeris time exist: Barycentric Dynamical Time (TDB) and Terrestrial Dynamical Time (TDT). ET and TDB are used synonymously in SPICE documentation. The TDB standard is used to describe the motion of celestial bodies relative to Solar System barycentre, while the TDT standard is used to describe the motion of bodies next to the Earth. These standard are linked by the relation :

$$\mathbf{TDB = TDT + 0.001657 \sin(E + 0.01671\sin(E))}$$

TDB is also linked with TAI by a constant values, in other words their difference is always 32.184 seconds:

$$\mathbf{TDB - TAI = 32.184\ s}$$

ET (or TDB) counts seconds past the reference epoch indicated with J2000 (approximately 1 January 2000, 12:00:00 at Greenwich).
For example, the precedent string

26 JULY 1986 1:30:07.162 (UTC)

correspond to

-424002537.65 seconds past the ephemeris epoch J2000

Most of spacecrafts has onboard clocks (**Spacecraft Clock**, SCL) to control time coverage of instruments. These clocks don't have linear time progress, so relations between SCLK, ET and UTC can't be described by linear functions.
Mission lifetimes are divided in several partitions where the clock works continuously. So time strings in spacecraft clocks are always preceded by the partition number, such as

1/4132564.034

where "1" is the partition number and the left numbers indicate the seconds of that partition.
Sometimes, in SPICE documentation the concept of **Julian Date** occurs to determine easily the number of days between two different epochs. This standard counts days and day fractions (in Julian Proleptic Calendar) past the noon (Greenwich time) of 1st January 4713 b.C.

Reference Frames

SPICE routines often ask users to choose in which reference frame the outputs have to be given. This choice is very important for the interpretations and later usage of outputs. SPICE supports several types of reference frames, such as:

- inertial reference frame
- body – fixed reference frame
- instrument – fixed reference frame

Inertial reference frames neither rotate or accelerate respect to fixed star. In these frames Newton's Laws are valid and they can be applied. SPICE usually uses J2000 coordinate system, where the Z-axis is aligned with Earth rotating axis, pointing in the direction of the north pole, the X-axis points in the vernal equinox direction (at J2000 epoch) and the Y-axis is defined so as to form a right-handed set of coordinate axis (as shown in figure 1.5).

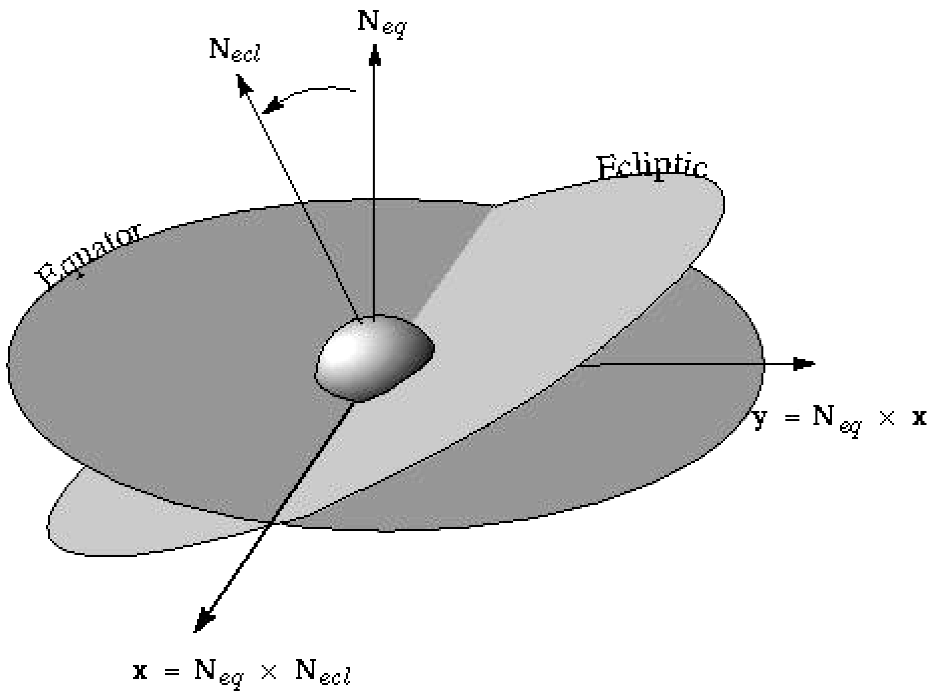

N_{eq}
N_{ecl}
Ecliptic
Equator
$\mathbf{y} = N_{eq} \times \mathbf{x}$
$\mathbf{x} = N_{eq} \times N_{ecl}$

Figure 1.5: Inertial reference frame. [11]

Body – fixed reference frames are tied with the surface of a body, centred at its centre and rotate respect to inertial frames. In SPICE documentation these coordinate systems are indentified with “IAU” prefix, since their orientation is determined by International Astronomical Union models. Dealing with Earth, the body - fixed reference frame is the Earth Centered Earth Fixed (ECEF), commonly defined in SPICE system as “IAU_EARTH”. This is a rotating frame centered in the mass center of the Earth, hence the name Earth-Centered. The z-axis is parallel to the Earth rotational axis pointing towards North. The x-axis intersect the sphere of the Earth at the 0° latitude, 0° longitude. This means that the ECEF rotates with the Earth around its z-axis. Therefore, coordinates of a point fixed on the surface don’t change, hence the name Earth-Fixed. The y-axis completes the right-handed frame (figure 1.6).

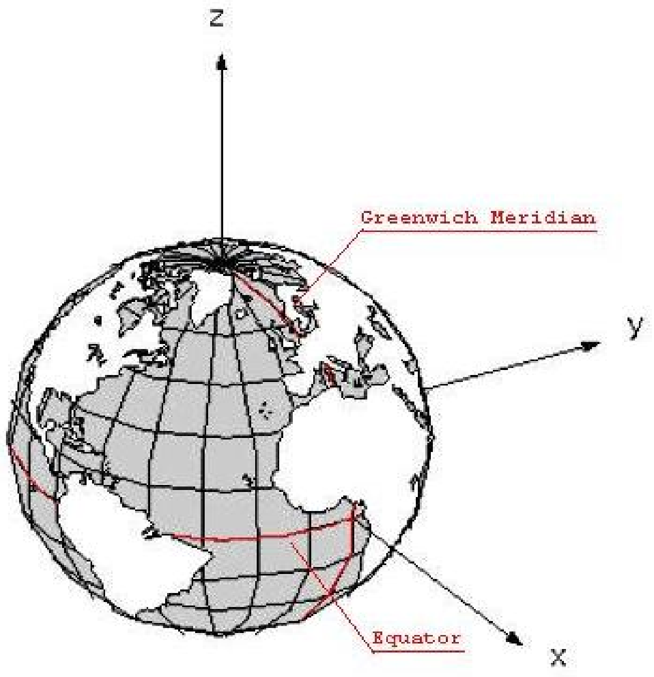


Figure 1.6: Earth Centered Earth Fixed Reference Frame. [11]

Instrument – fixed frames are tied with a specified instrument and are defined by the time – varying orientation of the instrument (or spacecraft).

Aberration Corrections

To determine accurately in which direction a remote sensing instrument must be pointed, or in which direction an antenna must be pointed to transmit a signal to a specified target, aberration corrections are needed. Within SPICE system, aberration corrections are adjustments made to accurately reflect the apparent state of a target body as seen from a specified observer at a specified time. Infact, in a pointing problem, for example, the instrument must point the apparent position of the target and not the real one at observation time. The real state is called "geometric" state. SPICE supports two aberration corrections: light time (called also planetary aberration) and stellar aberration.

Light time is the one-way light time between the position of target and the observer. **Light time aberration** correction is made determining where the target is when photons have been emitted. Light time correction only depends on motion between target and Solar System Barycentre (SSB), and it doesn't depend on velocity between observer relative to SSB. Figure 1.7 shown clearly light time aberration phenomenon.

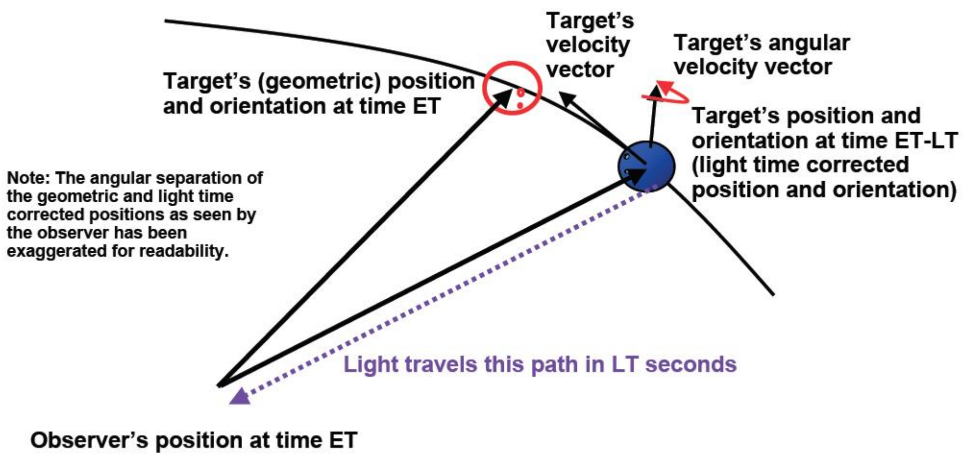

Target's velocity vector
Target's angular velocity vector
Target's (geometric) position and orientation at time ET
Target's position and orientation at time ET-LT (light time corrected position and orientation)
Note: The angular separation of the geometric and light time corrected positions as seen by the observer has been exaggerated for readability.
Light travels this path in LT seconds
Observer's position at time ET

Figure 1.7: Light Time Aberration phenomenon [11]. At time ET, the observer's camera records photons emitted from the target at time ET-LT. The camera sees the target's position and orientation at ET-LT.

Also observer velocity affects apparent target position: photons velocity relative to the observer is the difference between their velocity and observer velocity, always respect to SSB. This phenomenon is named stellar aberration and it doesn't depend on target velocity. Figure 1.8 shown clearly stellar aberration .

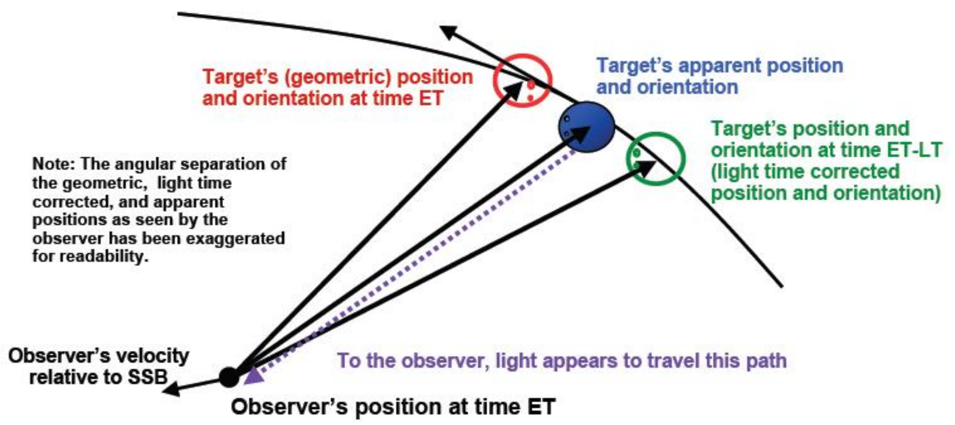

Target's (geometric) position and orientation at time ET
Target's apparent position and orientation
Target's position and orientation at time ET-LT (light time corrected position and orientation)
Note: The angular separation of the geometric, light time corrected, and apparent positions as seen by the observer has been exaggerated for readability.
Observer's velocity relative to SSB
To the observer, light appears to travel this path
Observer's position at time ET

Figure 1.8: Stellar Aberration phenomenon [11]. At time ET, the observer's camera records photons emitted from the target at time ET-LT. The vector from the observer at ET to the location of the target at ET-LT is displaced by a physical phenomenon called stellar aberration. The displaced vector yields the apparent position of the target.

According to application and usage, CSPICE routines allows to correct these aberrations.

Some applicative tags are:

1. NONE: routines return the geometrical target state, in other words without corrections;
2. LT: light time corrections applied;
3. LT+S: light time and stellar aberration corrections are both applied.

4. Introduction to MICE

MICE operates as an extension to the MATLAB environment. This environment includes an intrinsic capability to use external routines. MICE uses the MATLAB external interface functionality (MEX) to provide MATLAB users access to selected CSPICE routines from within MATLAB. A user need only install the interface library in order to take advantage of SPICE utilities. The MICE library contains the MATLAB callable C interface routines that wrap a subset of CSPICE wrapper calls. The wrapper files, named "*cspice_*.m*" and "*mice_*.m*", provide the MATLAB calls to the interface functions. The wrappers include a header section describing the function call, inputs-outputs (I/O) and examples, displayable by the MATLAB *help* command. These routines are *black boxes*, the algorithm existing behind routines is not visible and they are for NAIF staff use only. To make kernels available to SPACE programs, user has to load them. User can use the FURNSH routine to load all kernels-text and binary with the command:

>>cspice_furnsh ('name.ext')

It has to be noted that problems might raise if more than one kernel is loaded by the user.

Kernels are loaded into MATLAB session not into MATLAB scripts. This means that loaded kernels remain "active" throughout the whole MATLAB session. If there is only one script in the MATLAB session, there is no problem. On the other hand, some kernel data may be available and used to a script even though not intended to be so, driving to incorrect results. To bypass this problem there are two approaches:

1. include a call to *cspice_unload* for each kernel loaded using *cspice_furnsh* or include a call to *cspice_kclear* to remove all kernel data from the kernel pool loaded using *cspice_furnsh*;
2. load all needed kernels at the beginning of the session, paying careful attention to the files loaded and loading order. So user can create a filenames-list called "meta-kernel" and load the meta-kernel using FURNSH.

>>cspice_furnsh ('mykernels.furnsh')

This is a sample meta-kernel used to load a collection of kernels:

```
KPL/MK
\begindata
  PATH_VALUES    = ('/corista/mice/kernels')
  PATH_SYMBOLS   = ('KERNELS')
  KERNELS_TO_LOAD= (
            '$KERNELS/leapsecond.tls',
            '$KERNELS/sclk.tsc',
            'SKERNELS/de421.bsp')
```

The number of kernels that may be loaded at any time is large but limited (max 1000 for binary kernels and 1300 for all kernels). In the

following a detailed description of an applicative example is provided.
Use of MICE requires both the “lib” and the mice “src” directories existing in the MATLAB search path:

```
>>addpath (‘C: \corista\mice\lib\’)
>>addpath (‘C: \corista\mice\src\mice’)
```

To ensure a proper setup, execute the command

```
>>which mice
```

MATLAB should return the path to the mice.dll file. Writing

```
>>cspice_tkvrsn(‘toolkit’)
```

the command causes MATLAB to display the CSPICE library version (N0062).
MATLAB views all calls to MEX library as functions, that is a call as the form

output = name(inputs)

While for multiple arguments on output:

[output1, output2,…] = name (inputs)

This is a sample model to convert between several time system and time formats, built step by step. Through the cspice index, we learn the two routines of interest are **cspice_ str2et** and **cspice_ et2utc.**
cspice_ str2et converts a string representing an epoch to a double precision value representing the number of TDB seconds past the J2000 epoch corresponding to the input epoch.

```
  GIVEN     I/O   DESCRIPTION
---------   ---   ----------------------------------
   str       I    time string
    et      O    double precision number of TDB
                  seconds past the J2000 epoch that
                  corresponds to the input 'str'
```

cspice_et2utc converts an input time from ephemeris seconds past J2000 to Calendar, Day-of-Year, or Julian Date format, UTC.

```
  GIVEN     I/O   DESCRIPTION
---------   ---   ----------------------------------
   et        I    double precision array of
                  ephemeris time
  format     I    format flag describing the output
                  time string
  prec       I    number of decimal of precision
  utcs       O    output string
```

Format flag are:

```
  'C'      Calendar format, UTC
  ‘D'      Day-of-Year format, UTC
  'J'      Julian Date format, UTC
  'ISOC'   ISO Calendar format, UTC
  'ISOD'   ISO Day-of-Year format, UTC
```

Building the program as an *.m file:

```
%***********************************************************
%****** This program converts UTC time format in:     ******
%****** - Et format                                   ******
%****** - Calendar format, UTC                        ******
%****** - Day-of-Year format, UTC                     ******
%****** - Julian Date format, UTC                     ******
%***********************************************************
%load leapseconds file
cspice_furnsh ('C:\Corista\mice\lib\naif0009.tls');
%input
time = input( 'Enter time string => ','s');
%conversion UTC==>ET
et = cspice_str2et( time );
%conversion ET==>UTC "C"
```

```
UTCC = cspice_et2utc( et, 'C', 6 );
%conversion ET==>UTC "D"
UTCD = cspice_et2utc( et , 'D', 6 );
%conversion ET==>UTC "J"
UTCJ = cspice_et2utc( et , 'J', 6 );
%output
fprintf( 'ET==> %11.6f\n',et );
fprintf( 'Calendar format ==> %s\n', UTCC );
fprintf( 'Day-Of-Year format ==> %s\n', UTCD );
fprintf( 'Julian Date format==> %s\n', UTCJ );
```

5. SPICE Ephemeris Subsystem SPK

An SPK file contains ephemeris data for "ephemeris objects". Spacecrafts, planets, satellites, comets and asteroids are the obvious kinds of ephemeris objects, but many other possibilities exist, such as:

- a rover on the surface of a body
- a camera on top of a mast on a lander
- a transmitter cone on a spacecraft
- a deep space communications antenna on the earth
- the center of mass of a planet/satellite system (planet barycenter)
- the center of mass of our solar system (solar system barycenter)

All possibilities are summarised in the following picture (fig. 1.9)

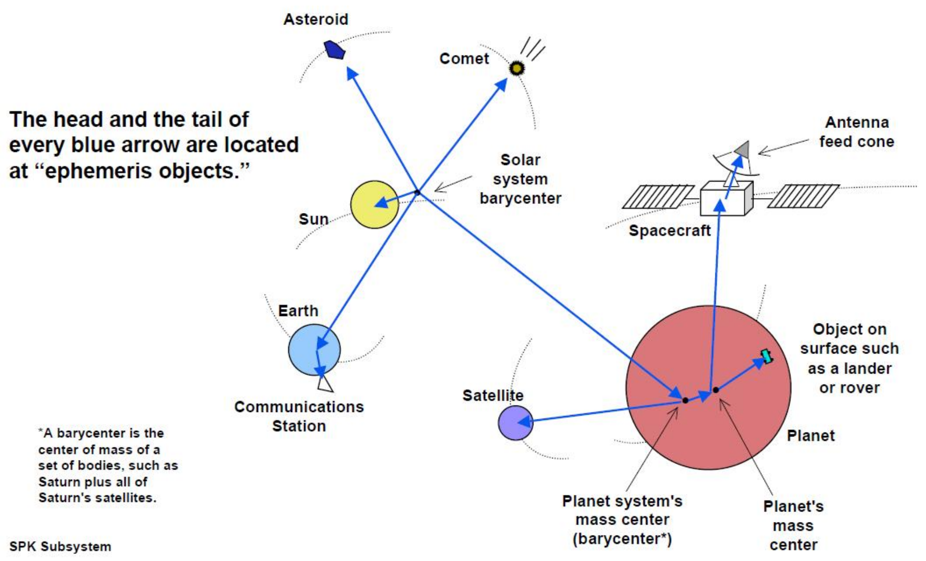

Asteroid
Comet
The head and the tail of every blue arrow are located at "ephemeris objects."
Solar system barycenter
Antenna feed cone
Sun
Spacecraft
Earth
Object on surface such as a lander or rover
Communications Station
Satellite
Planet
*A barycenter is the center of mass of a set of bodies, such as Saturn plus all of Saturn's satellites.
Planet system's mass center (barycenter*)
Planet's mass center
SPK Subsystem

Figure 1.9: Example of ephemeris objects. [11]

Inside an SPK file ephemeris objects come in pairs: a "body" and a "center of motion" so that the ephemeris is given for the body moving relative to the center of motion.

For the position component, the vector points TO the body FROM the center of motion. In the case of multiple pairs, reading an SPK file, user has to specify which ephemeris object is the "target" and which is the "observer". Other used conventions are:

- the position data point from the "observer" to the "target";
- the velocity is of the "target" relative to the "observer".(Fig.1.10)

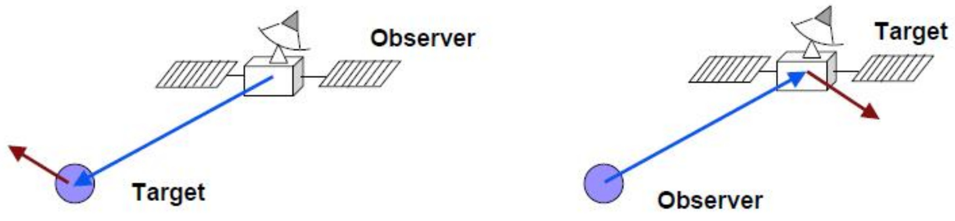


Figure1.10: Example of conventions. [11]

An important parameter concerning with time is the “coverage” or “time coverage”. Coverage is the time period over which an SPK file provides data for an ephemeris object.

Since SPK kernels are binary files, reading informations and data inside is not easy, so that it’s impossible to open such files with a traditional text editor. NAIF provides making available several utility program (including in the toolkit) to bypass this problem. The program BRIEF, for example, allows users to read general informations and time coverage of SPK kernels. But BRIEF is not the only one. Launching SPACIT and choosing the option “S” (summarize binary file), the program displays the “descriptor” of an SPK kernel.

Concerning with binary files, porting kernels may cause several problems. Data formats vary across platforms, so data files created on a platform may not be useful on another platform (called “incompatible” platforms): different platforms use different bit patterns to represent numbers (and possibly characters). This problem not only affects binary data formats but also text formats: different platforms may use different mechanisms to represent “lines” in text files. Platforms have “compatible” binary or text formats if they use the same binary or text data representations. NAIF with toolkit utility programs solves the porting problem. SPACE toxfr.exe and spacit.exe may be used to convert binary data format kernels in a “transfer format”. Later on, after porting, spacit.exe and tobin.exe allow to reconvert the transfer format file in a binary data format file, available and ready-to-use on the new platform.

SPK kernels can be downloaded from: ftp://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk .

There are four sub directories concerning with comets, planets, satellites, and asteroids. Our attention is focused on the SPK planets kernel: "de421.bsp", realised on 31st March 2008.

To extract position or state vectors of ephemeris objects from an SPK file, usually two kinds of SPICE kernels are needed:

- SPK, ephemeris kernels, sometimes just one is needed;
- LSK, leapseconds kernel, used to convert between Coordinated Universal Time (UTC) and Ephemeris Time (ET); usually needed since most people work with UTC time.

To retrieve state vectors of an ephemeris object, *cspice_spkezr* may be used. This routine returns the state (position and velocity) of a target body relative to an observing body, optionally corrected for light time (planetary aberration) and stellar aberration. The call is:

[state, lt] = cspice_spkezr (targ, et, ref, abcorr, obs)

Inputs:

- **targ, obs**: characters names or NAIF IDs for the target body and the observer one, in other words the point and the origin of the state vector (Cartesian position and velocity) to be returned;
- **et**: the time at the observer at which the state vector is to be computed in Ephemeris Time;
- **ref**: the scalar name of the reference frame relative to which the output state vector should be expressed;
- **abcorr**: kind of aberration correction(s) to be applied.

Outputs:

- **state**: Cartesian state vector with six components: three for position and three for velocity of the target respect to the observer;
- **lt**: the one-way light time between the position of target (optionally aberration corrected) and the geometric position of the observer at the specific epoch.

Building the program as an *.m file:

```
%***********************************************************
%****** This program retrieves state vectors         ******
%****** (position and velocity) of a target body      ******
%****** relative to an observing body optionally      ******
%****** corrected for light time at a specific epoch  ******
%***********************************************************
%initial conditions
cspice_kclear
%load leapseconds file and SPK kernel
cspice_furnsh( 'C:\Corista\mice\lib\naif0009.tls');
cspice_furnsh( 'C:\Corista\mice\lib\de421.bsp');
%input
time= input ('Enter the time string => ','s');
targ= input ('Enter the target body => ','s');
obs= input ('Enter the observer body => ','s');
frame= input ('Enter the reference frame => ','s');
abcorr= input ('Enter the aberration corrections => ','s');
%time conversion between string => ET
et = cspice_str2et( time );
%compute state vectors
[pos,ltime ]=cspice_spkezr( targ, et, frame, abcorr, obs );
%outputs
fprintf( 'Position (km) x : %11.6f\n', pos(1) );
fprintf( 'Position (km) y : %11.6f\n', pos(2) );
fprintf( 'Position (km) z : %11.6f\n', pos(3) );
fprintf( 'Velocity x (km/s) : %11.6f\n', pos(4)  );
fprintf( 'Velocity y (km/s) : %11.6f\n', pos(5)  );
fprintf( 'Velocity z (km/s) : %11.6f\n', pos(6)  );
fprintf( 'ET : %11.6f\n', et )
fprintf( 'Light Time : %11.6f\n', ltime )
```

The planetary and lunar ephemeris DE 421 represents the “current best estimates” of the orbits of the Moon and planets. The lunar orbit is currently known to sub-meter accuracy though fitting lunar laser ranging data. The orbits of Venus, Earth, and Mars are known to sub-kilometer accuracy. Because of perturbation of the orbit of Mars by asteroids, frequent updates are needed to maintain the current accuracy into the future decade. The orbits of Earth and Mars are continually improved through measurements of spacecraft in orbit about Mars. Mercury's orbit is determined to an accuracy of several kilometers by radar ranging. The orbits of Jupiter and Saturn are determined to accuracies of tens of kilometers as a result of spacecraft tracking and modern ground-based astrometry. The orbits of Uranus, Neptune, and Pluto are not as well determined.

The axes of ephemeris are oriented with respect to International Celestial Reference Frame (ICFR). For DE 421 the positions of the Moon and planets were integrated using a n-body parameterized post-Newtonian metric (PPN). The PPN parameters γ and β have been set to 1, their values in general relativity. The oblateness of the Sun has been modelled with J_2 set to $2.0x10^{-7}$. Along with the Earth-Moon mass ratio, the mass parameter GM for the Sun, which is by convention a fixed value in units of AU^3/day^2, was estimated in units of km^3/s^2 by solving for the AU in km in the development of DE 421. The mass parameter of the Earth-Moon system was held fixed to a previous LLR-only estimate. The mass parameters for the other planets (planetary systems for planets with natural satellites) were taken from published values derived from spacecraft tracking data.

For the Earth's gravity field, J_3 and J_4 were taken from the GGM02C gravity field and the equatorial Earth radius used with gravity was set to 6378.1363 km. The J_2 coefficient was based on the GGM02C "tide free" value, but the J_2 value was adjusted with different Love numbers. Love numbers, introduced by A.E.H. Love, determine the ratio of the height of a body tide to the static marine tide and the ratio of additional potential produced by the redistribution of mass to the displacement of the crust to that of the equilibrium fluid tide.

It’s proper to spend few words about GGM02. GGM02S gravity model was estimated with 363 days (spanning form April 2002 to March 2003) of GRACE K-band range rate, attitude and accelerometric data. No satellite information, or surface gravity information, or other a priori conditioning were applied in generating this solution. GGM02C, a high resolution global gravity model, combines GGM02S with terrestrial gravity information (surface gravity and mean sea surface). The Earth tides came from the IERS conventions and the ocean tides were based on FES2004 (Finite Element Solution tidal atlates).

➢ CHAPTER II: Kernels generation

From NAIF server are available kernels for several missions. SPICE is used essentially for all NASA solar system exploration projects. SPICE kernels archived by NAIF node deals both with passed (VOYAGER, PHOBOS,…), current (CASSINI, MESSENGER,…) and future missions (MSL, JUNO, …). Limited SPICE kernels may be created (or are being) for some past missions: for example for some missions only SPK is available. SPICE is also used in support of some space physics and astrophysics missions (HST,…), on some non-NASA missions (HAYABUSA, ROSETTA,…).
Missions (and relative kernels) archived and available on NAIF server are listed in Annex I.
NAIF also provides tools to generate your own kernels taking high advantage from SPICE routine. According to their requirements, users can create the kernels they need for their computations and analysis. Even thought users can create kernels for their needs, NAIF's tools don't allow to create each type of kernels: some tools have limitations and some data indicating how to create a complete set of kernels are not available.
This work is focused on creating kernels in the scenario of Post-EPS mission, in particular retrieving Sun and Moon view angle with respect to the spacecraft and the coverage analysis.

1. Post–EPS mission

EUMETSAT Polar System (EPS) mission is the first European polar orbiting operational meteorological satellite system and it's part of the Global Operational Satellite Observation System (GOS), as shown in fig. 2.1.

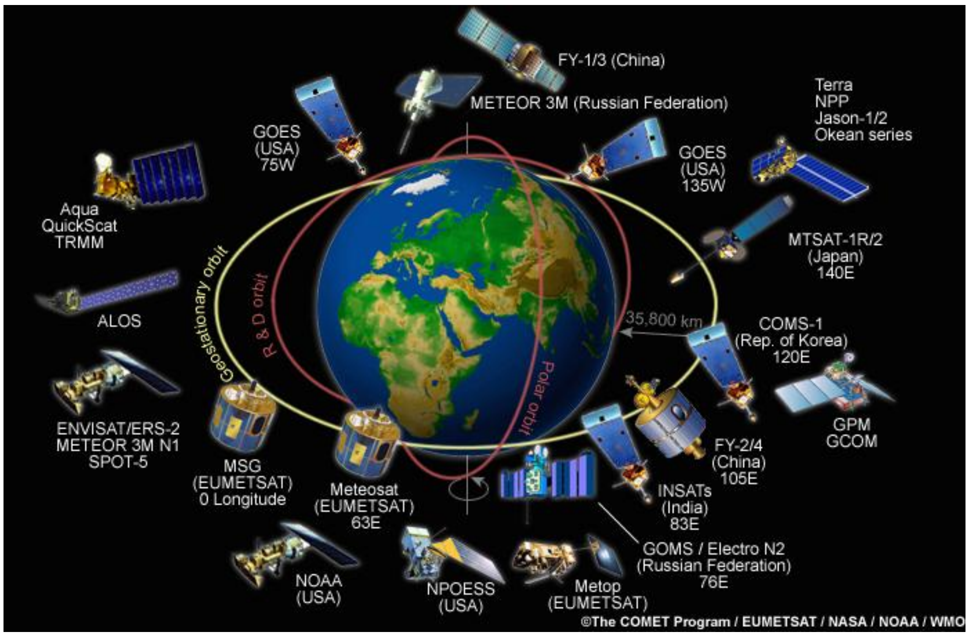


Figure 2.1: EPS contributes to the GOS. [5]

EPS is the European contribution to a joint satellite system between Europe and United States, called Initial Joint Polar-Orbiting Operational Satellite System (IJPS). This is an agreement between the European Organisation for the Exploration of Meteorological Satellites (EUMETSAT) and the National Oceanic an Atmospheric Administration (NOAA). The EPS programme consists of a series of three Meteorological Operational (MetOp) satellites, operative for more than 14 years from 2006. MetOp-2, the first satellite, was launched on 19 October 2006 from the Baikonur cosmodrome in Kazakhstan with a Soyuz launcher. Once in orbit the satellites are ordered alphabetically, so the first satellite that was launched is called MetOp-A. Each satellite has a nominal lifetime in orbit of 5 years. This programme has brought a new era in the observations of Earth's weather, climate and environment, and it will significantly improve operational meteorology, in particular Numerical Weather Prediction (NWP). The data carried by MeteOp can be assimilated directly into NWP models to compute forecasts ranging from a few hours up to ten days ahead. Measurements from microwave radiometers on board Metop provide NWP models that retrieve very important informations about global atmospheric temperature an humidity structure, with a high vertical and horizontal resolution.

Post- EPS mission is a mandatory programme which is the extension of EPS observation missions, focusing in particular on NWP and climate monitoring starting in 2018 for at least 15 years (figure 2.2).

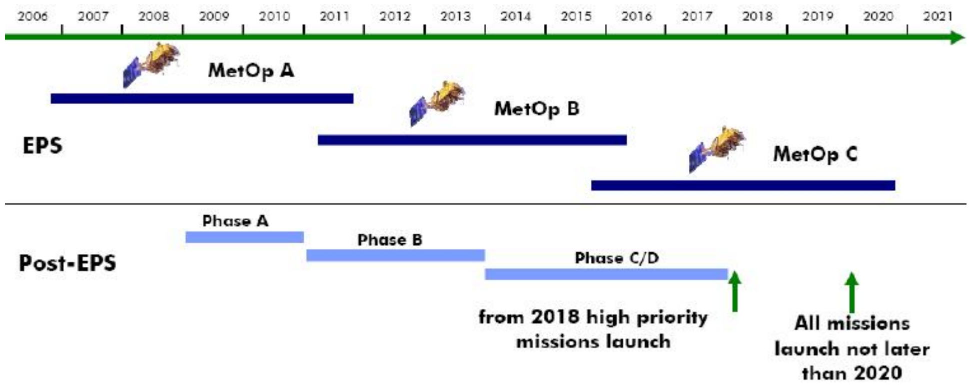


Figure 2.2: Post-EPS mission plan. [13]

Post-EPS orbits are Sun-Synchronous Orbit (SSO) with the orbit geometry shown in figure 2.3:

- Altitude, defined as (Mean Semi-Major Axis – Earth equatorial radius): km 720;
- SSO at 9:30 LTDN;
- Angle between Earth-Sun direction (at equinox) and lines of node:37.5°
- Angle Earth-Sun direction-orbit plane: 38.7° (max), 27.7° (min), 32.8° (mean).

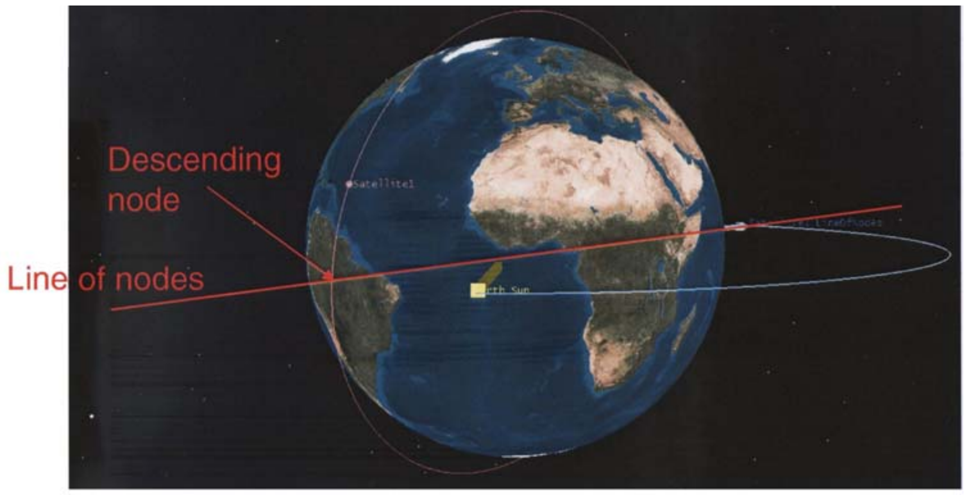


Figure 2.3: Post-EPS orbit geometry. [13]

Sun-synchronous orbits are geocentric orbits where a satellite passes over any given point of the Earth's surface at the same local time. The surface illumination angle will be nearly the same every time. This means that line of nodes and Sun projection onto the equatorial plane have the same precession rates, i.e. angle between the Earth-Sun direction equatorial projection and the line of nodes is constant. For SSO, combining altitude, inclination and eccentricity the following equations can be written:

$$\dot{\Omega}(a,e,i) = \dot{\alpha}_{\Theta}$$

$$-\frac{3}{2}J_2\left(\frac{R}{a(1-e^2)}\right)\sqrt{\frac{\mu}{a^3}}\cos i = \dot{\alpha}_{\Theta}$$

Where:

- $\dot{\Omega}$ is the line of nodes precession rates;
- $\dot{\alpha}$ is the sun precession rates;
- J_2 is the is the second zonal harmonic coefficient (Earth oblateness);
- R is the Earth equatorial radius;
- a is the semi-major orbit axis;
- e is the eccentricity;
- i is the orbital inclination;
- μ is the gravitational parameter.

2. Making kernels

In order to compute illumination angles of the Post-EPS satellite and analyze the coverage, the user needs an SPK containing the spacecraft ephemeris to retrieve time by time the state vector. Then, with respect to a fixed spacecraft reference system, the illumination angles and the coverage can be easily computed.
The user must provide the SPICE system the required kernels to make the analysis and gain from SPICE routines. The figure 2.4 shows a logical overview of the analysis developed in this chapter.

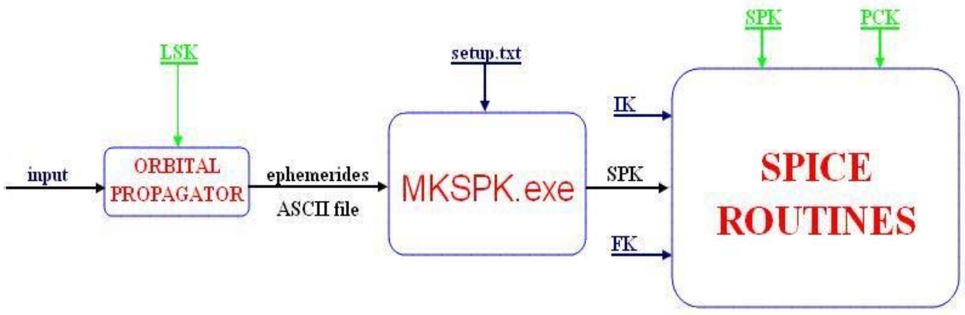


Figure 2.4: Logical summary of kernels generation. SPICE routines requires several kernels: SPICE kernels (in green) provided by SPICE itself, text kernels and setup file (in blue) , and satellite's ephemeredes (in black) created in order to conduce the analysis.

Four main steps are needed:

- create the SPK;
- create a FK;
- create an IK;
- finally write the MICE program.

In the following these four step will be described in detail.

Making an SPK file

Before creating an SPK, it is useful to understand and to know the structure of an SPK file. An SPK file is made up of one or more data "segments" and a "comment" area (figure 2.5).

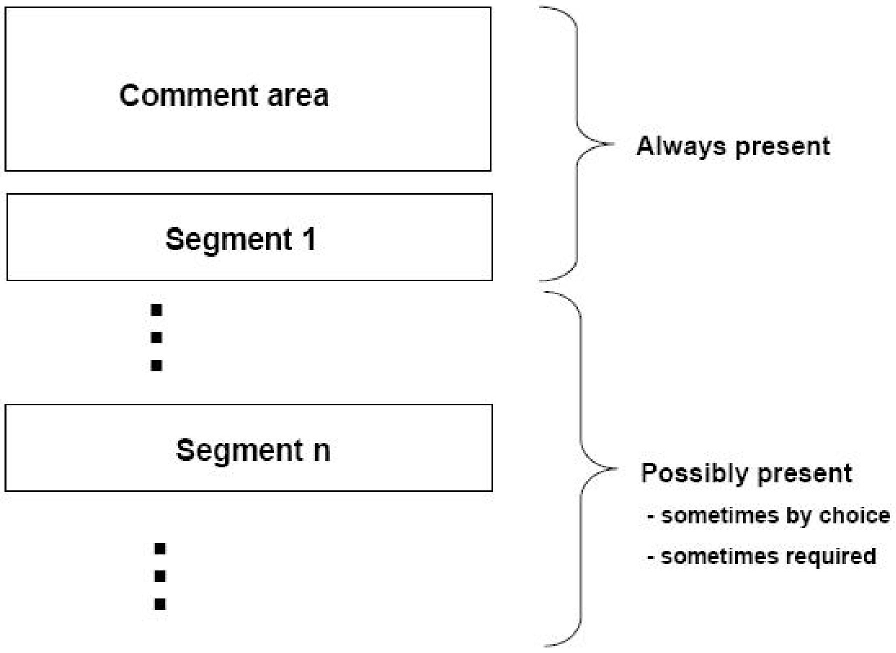


Fig. 2.5: Logical Organization of an SPK file. [11]

Each segment contains ephemeris data sufficient to compute the geometric state (position and velocity) of one solar system body (the target) with respect to another (the center) at any epoch throughout some finite interval of time. The space body may be a spacecraft, a planet or planet barycenter, a satellite, a comet, an asteroid, a tracking station, a roving vehicle, as well as an arbitrary point for which an ephemeris has been calculated. Each body in the solar system is identified by a unique integer code. The states computed from the ephemeris data in a segment must be referenced to a single, recognized reference frame.

The summary for each segment (figure 2.4), called "descriptor", retrieves:

- the segment's name;
- the NAIF integer code for the target;
- the NAIF integer code for the center;

- the NAIF integer code for the reference frame;
- the integer code for the representation (type of ephemeris data);
- time coverage;

  The fifth integer component of the descriptor - the code for the representation, or "data type" - is the key to the SPK format: it describes how ephemeris data are represented inside the SPK file. Each type has certain properties that may promote or limit its usefulness in a particular application. SPICE currently support 18 data types:

1. Modified Difference Arrays (MDA).

   These are primarily used for spacecraft ephemeredes. Each segment containing Modified Difference Arrays contains an arbitrary number of logical records. Each record contains difference line coefficients valid up to a final epoch. A given function contains the algorithm used to construct a state from a particular record and epoch. Each one of these records contains 71 double precision numbers.

1. Chebyshev polynomials (position only).

   These are sets of coefficients for the x, y, and z components of the body position. The velocity of the body is obtained by differentiation. This data type is normally used for planet barycenters, and for satellites whose orbits are integrated.

2. Chebyshev polynomials (position and velocity).

   These are sets of coefficients for the x, y, and z components of the body position, and for the corresponding components of the velocity. This data type is normally used for satellites whose orbits are computed directly from theories.

3. Reserved for future use.

4. Discrete states (two body propagation).

   This data type contains discrete state vectors. A state is obtained for a specified epoch by propagating the state vectors to that epoch according to the laws of two body motion and then taking a weighted average of the resulting states.
5. Reserved for future use.
6. Reserved for future use.
7. Equally spaced discrete states (Lagrange interpolation).

   This data type contains discrete state vectors whose time tags are separated by a constant step size. A state is obtained for a specified epoch by finding a set of states "centered" at that epoch and using Lagrange interpolation on each component of the states.
8. Unequally spaced discrete states (Lagrange interpolation).

   This data type contains discrete state vectors whose time tags may be unequally spaced. A state is obtained for a specified epoch by finding a set of states "centered" at that epoch and using Lagrange interpolation on each component of the states.
9. Space Command Two-line Elements (Short Period Orbits).

   This data type contains Space Command two-line element representations for objects in Earth orbit.
10. Reserved for future use.
11. Hermite Interpolation Uniform Spacing.
12. Hermite Interpolation Non-uniform Spacing.
13. Chebyshev polynomials non-uniform spacing.

    This data type contains Chebyshev polynomial coefficients for the position and velocity of an object. Unlike SPK Types 2 and

3, the time intervals to which polynomial coefficient sets apply do not have uniform duration.

14. Precessing conic propagation.
    This data type allows for first order precession of the line of apsides and regression of the line of nodes due to the effects of the J2 coefficient in the harmonic expansion of the gravitational potential of an oblate spheroid.
15. Reserved for future use.
16. Equinoctial Elements.
    This data type represents the motion of an object about another using equinoctial elements. Unlike Type 15, the mean motion, regression of the nodes and precession of the line of apsides are not derived from the gravitational properties of the central body, but are empirical values.
17. Hermite/Lagrange Interpolation.

After the description of the structure of an SPK, it's now possible to analyze the method available for making an SPK file.

NAIF provides a conversion utility (**MKSPK.exe**) that takes a data file produced by an orbital propagator as input and gives the binary files (figure 2.6).

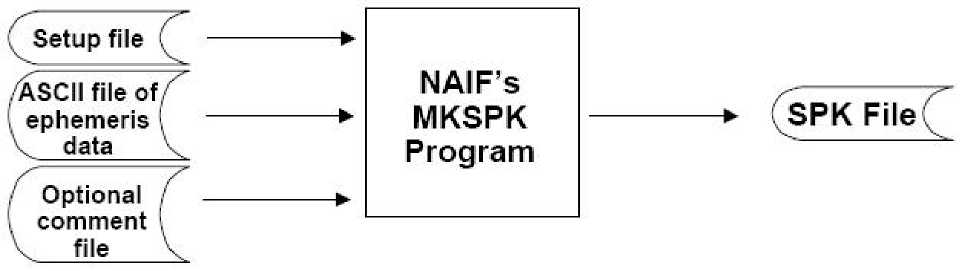


Fig. 2.6: Logical diagram for making SPK. [11]

As shown in figure 2.5, a setup file and an ASCII file of ephemeris data are required (a comment file is optional). MKSPK doesn't allow to create SPK files in whatever data type. The needed ephemeris data representation are:

- table of Cartesian state vectors;
- one or more sets of Space Command two-line;
- table of conic elements;
- one or more sets of equinoctial elements.

The possible SPK data type produced are 5, 8, 9, 10, 12, 13, 15, 17 (see list above).

The setup file (figure 2.5) provides the conversion utility to read and storage the ephemeris data file, describing the observer, the target, the reference frame, the type of input and output data, how ephemeredes are placed in the data file. The format of this file must be conform to the SPICE text kernel specification. This means that the input values must be assigned to keyword variables through the format:

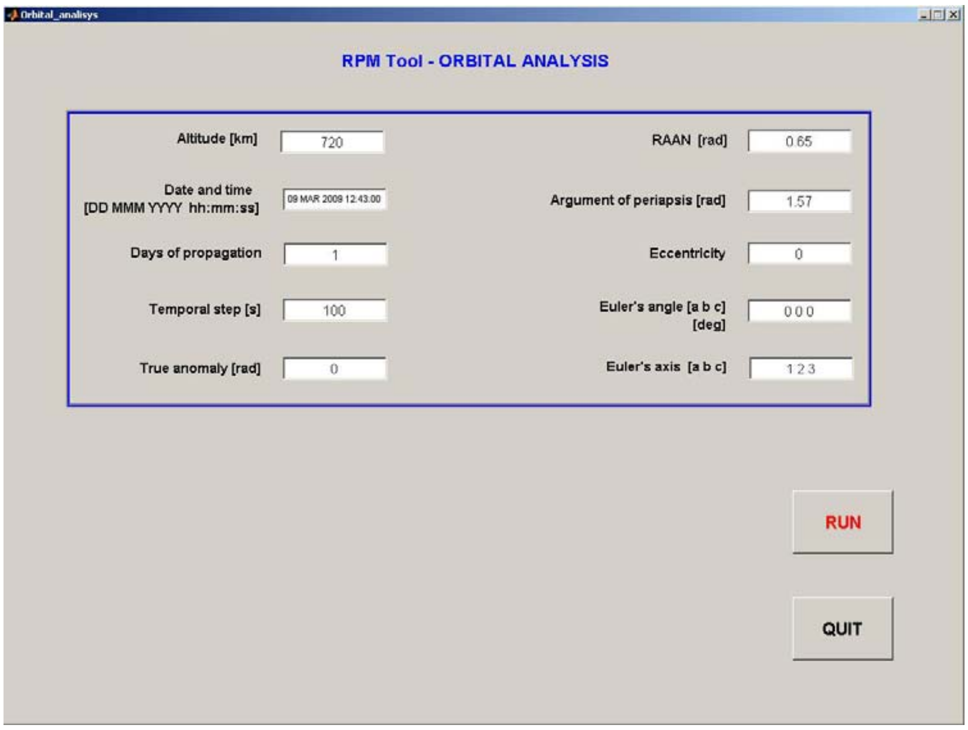
```
KEYWORD = VALUE
```

The names of keywords must be strictly uppercase while the value of keywords don't matter if it's upper, lower or mixed case. Each assignment is restricted to a single line and sets of this assignment must be enclosed between

```
\begindata
\begintext
```

Comments can be written before `\begindata` and after `\begintext.`

The assignments required for the setup file are listed in Annex II. Compiled the setup file, an ephemeris data file is needed. The ephemeris data are produced by an orbital propagator (not provided by NAIF). The orbital propagator used in this thesis is a trajectory propagator, which takes advantage of the power of NAIF library and the fast and the high level computation of SPICE routines. The algorithm used take into account the Earth oblateness (J_2 term) in scenario of Post-EPS mission which satellites lies on LEO orbits. The propagator considers that the true anomaly at the initial propagation time is zero and gives to the user the possibility to set. Fixed, the set of orbital parameters, the user can choose the initial date of propagation, the days of propagation and the time step (figure 2.7).



RPM Tool - ORBITAL ANALYSIS

Altitude [km]	720	RAAN [rad]	0.65
Date and time [DD MMM YYYY hh:mm:ss]	09 MAR 2009 12:43:00	Argument of periapsis [rad]	1.57
Days of propagation	1	Eccentricity	0
Temporal step [s]	100	Euler's angle [a b c] [deg]	0 0 0
True anomaly [rad]	0	Euler's axis [a b c]	1 2 3

Figure 2.7: Orbital propagator user interface.

The software, using SPICE routines, converts the time string in Ephemeris Time, and computes the orbit inclination for a Sun-Synchronous Orbit using the formula considered above. Beside this other outputs of the orbital propagator are: the mean anomaly, M, and the eccentric anomaly, E, solving the transcendent Kepler equation using the Newton-Raphson iteration method.
The equation that links mean and eccentric anomaly is:

$$M = n(t - T) = E - e \sin E$$

where n is defined as $n = \sqrt{\mu / a^3}$ called "mean motion". The relationship between eccentric anomaly and true anomaly ν and radius r is:

$$\cos \nu = \frac{e - \cos E}{e \cos E - 1}$$

$$\sin \nu = \frac{a\sqrt{1 - e^2}}{r} \sin E$$

$$r = a(1 - e \cos E)$$

Retrieved radius and true anomaly, the components of position and velocity of the satellite for the considered time frame are computed in ECI reference. Considering Earth rotation and the related transformation matrix, the algorithm converts state components from ECI to ECEF reference. The compiled program file (*.m file) of the orbital propagator is reported in Annex III.
Ephemeredes in double precision are printed in a text file. The orbital propagator creates a time ordered sets of states, where each

ephemeris epoch lies one on a single line and components are separated by TAB: the first value is the epoch in Ephemeris seconds, followed by three position components and three velocity components. A sample of the ephemeris data file is the following:

```
ephemeris [et,state].Tar: -313 Obs: EARTH
 287712066.19 7098.14   0.00    0.00  0.00 -1.08 7.42
 287712076.19 7097.73 -15.95   74.16 -0.08 -1.08 7.42
 287712086.19 7096.52 -31.91  148.30 -0.16 -1.08 7.41
```

For a more comfortable visualization according to this page format, the TAB has been replaced with SPACE and the values has been cut at the second decimal number.

Created the ASCII ephemeris data file, the user can compile the setup file to provide the MKSPK conversion program to read in the right way the ephemeredes and give the binary SPK files. Assumed that the considered satellite ID is "-313" (a fictitious number chosen by user), the setup file will be:

```
\begindata
INPUT_DATA_FILE   = 'ephemerisdata.txt'
OUTPUT_SPK_FILE   = 'eph.bsp'
INPUT_DATA_TYPE   = 'STATES'
OUTPUT_SPK_TYPE   = 5
OBJECT_ID         = -313
CENTER_NAME       = 'EARTH'
REF_FRAME_NAME    = 'IAU_EARTH'
PRODUCER_ID       = 'SALVATORE TUOSTO'
DATA_ORDER        = 'EPOCH X Y Z VX VY VZ'
INPUT_DATA_UNITS  = ('ANGLES=DEGREES'
                     'DISTANCES=km')
DATA_DELIMITER    = 'TAB'
LINES_PER_RECORD  = 1
IGNORE_FIRST_LINE = 2
LEAPSECONDS_FILE  = 'naif0009.tls'
PCK_FILE          = 'Gravity.tpc'
```

```
TIME_WRAPPER      = '# ETSECONDS'
SEGMENT_ID        = 'PROVA_SPK_MIO_SAT'
APPEND_TO_OUTPUT  = 'YES'
\begintext
```

The required files are now available and ready to be given in input to MKSPK conversion program.

With the created SPK file, user can retrieve position and velocity of the satellite at any epoch enclosing in time coverage using SPICE routines analyzed in the past chapter (Ch. 1.4). The satellite's SPK must be loaded as a generic SPK file with the FURNSH routine.

Making an FK (Frame Kernel) file

For pointing problem, or simply to compute positions of several celestial bodies with respect to a spacecraft, a rover, an orbiter, a specified instrument, defining a body reference frame is a crucial problem. A number of reference frames are already defined in SPICE system, but sometimes a specific problem can require the creation of a new reference frame. In order to define a new reference frame, its orientation and position with respect an existing reference frame have to be computed: SPICE allows also to compute transformation between neighboring reference frame.

To create a Frame Kernel (FK file), the concept of a frame class is needed. The method by which a frame is related to some other frame is a function of the "class" of the frame. There are five classes:

1. Inertial frame.

   These frames don't rotate with respect to the star background. Here, Newton's laws can be applied. The class number associated with inertial frame is 1.

2. Body-Fixed (PCK) frames.

   The orientation of these frames with respect to inertial frames is supplied in PCK files. The class number associated with PCK frames is 2.
3. CK frames.

   The orientation of these frames with respect to some other reference frame is supplied via a C-kernel. A CK file holds orientation data or a moving structure on the spacecraft. C-kernels use spacecraft clock, so user must load a SCLK file appropriate for the C-kernel. The class number associated with CK frames is 3.
4. Fixed offset frames.

   These frames have a constant orientation with respect to some reference frame an this orientation is included in a SPICE text kernel. That’s why this class of frame is also called Text Kernel (TK). The class number associated with TK frames is 4.
5. Dynamic frames.

   These frames time-dependent, and they are defined via parameters or formulas specified in a text frame kernel. The class number associated with dynamic frames is 5.

   Dynamic frames are time dependent so they are ideal to describe orbital frames of any ephemeris object, also because the frames of the fifth class are easy to define and enables SPICE system to use conveniently a wide variety of frame that are not “built-in” to SPICE. The only currently frame definition style supported by dynamic frames subsystem is the parameterized one. A “parameterized dynamic frame” is defined by a formula implemented in SPICE code and having selectable parameters set via a frame kernel. The parameterized dynamic frame “family” indicates the mathematical formula the frame is defined by. There are five parameterized dynamic frame families:

- **Two-vector frames**: these reference frames are defined by two vectors. The first vector is parallel to one axis of the frame; the component of the second vector orthogonal to the first is parallel to another axis of the frame, and the cross product of the two vectors is parallel to the remaining axis.
- **Mean equator and equinox of date frames**: these use mathematical precession models to define orientation of a body's equatorial plane and location of the frame's x-axis. Currently these frames are supported only for the earth.
- **True equator and equinox of date frames**: these use mathematical precession and nutation models to define orientation of a body's equatorial plane and location of the frame's x-axis. Currently these frames are supported only for the earth.
- **Mean ecliptic and equinox of date frames**: these use mathematical precession and mean obliquity models to define orientation of a body's orbital plane and location of the frame's x-axis. Currently these frames are supported only for the earth.
- **Euler frames**: polynomial coefficients, a reference epoch, and an axis sequence are used to specify time-dependent Euler angles giving the orientation of the frame relative to a second, specified frame as a function of time.

In order to create a reference frame fixed to a spacecraft it's comfortable to adopt the two-vectors frame family: user can choose how to orient spacecraft frame with respect to existing vectors in the easiest and most useful way.

Two-vector frames use two user-specified, non-parallel vectors to define the mutually orthogonal axes of a right-handed reference frame. In these frames, one vector is parallel to a specified axis of the reference frame: this vector is called the “primary vector”. The other one, called the “secondary vector” defines another axis: the component of the secondary vector orthogonal to the primary vector is parallel to a specified axis of the reference frame. Each suitable vector may be:

- position vector, defined by the position of one ephemeris object respect to another;
- target near point vector, defined by as the vector from an observer to the nearest point on a specified extended target body to that observer;
- velocity vector, defined by the velocity of an ephemeris target object relative to an observing ephemeris object;
- constant vector, defined as a vector constant in a frame specified by the kernel creator.

Figure 2.8 shows graphically how to use and define two vectors frames in an applicative example.

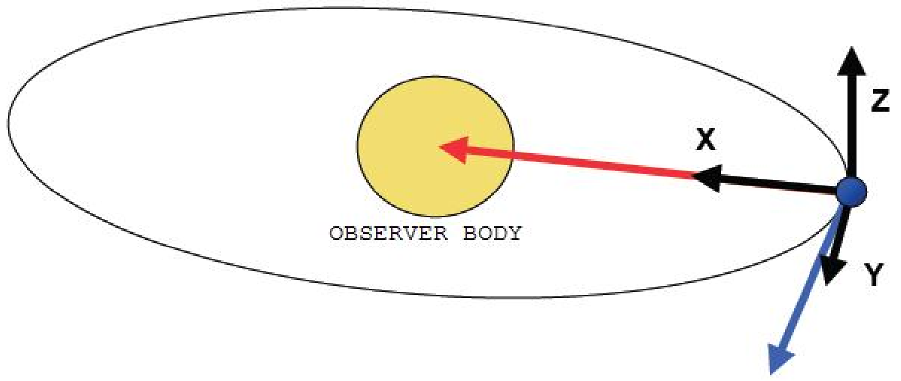


Figure 2.8: Two vectors frame. The primary vector (the red one) is defined by the position of the target body (the small one) respect to the observer body. The secondary vector (the blue one) is defined as the velocity of the target body. The X-axis is associated with the primary vector, the Y-axis with the normalized component of the secondary vector orthogonal to primary vector and the Z-axis completes the right-handed frame .[3]

Finally to define a new reference frame, user have to create a frame kernel. The format of this file must be conform to the SPICE text kernel specification, in other words it has to follow the same rules of SPK setup file viewed above. The assignments to define a two vectors frame kernels are reported in Annex IV.

The new reference frame associated with the considered satellite (-313) is “Donald” (a fictitious name), and its ID code is 961934 (a fictitious number). It is a two vectors reference frame, where the primary vector is the position vector of the satellite respect to the Earth, and the secondary vector is the satellite velocity respect to the Earth. The frame kernel will be:

```
\begindata
     FRAME_DONALD                    = 961934
     FRAME_961934_NAME               = 'DONALD'
     FRAME_961934_CLASS              = 5
     FRAME_961934_CLASS_ID           = 961934
     FRAME_961934_CENTER             = -313
     FRAME_961934_RELATIVE           = 'IAU_EARTH'
     FRAME_961934_DEF_STYLE          = 'PARAMETERIZED'
     FRAME_961934_FAMILY             = 'TWO-VECTOR'
     FRAME_961934_PRI_AXIS           = '-Z'
     FRAME_961934_PRI_VECTOR_DEF =
                        'OBSERVER_TARGET_POSITION'
     FRAME_961934_PRI_OBSERVER   = 'EARTH'
     FRAME_961934_PRI_TARGET     = -313
     FRAME_961934_PRI_ABCORR     = 'NONE'
     FRAME_961934_PRI_FRAME      = 'IAU_EARTH'
     FRAME_961934_SEC_AXIS       = 'Y'
     FRAME_961934_SEC_VECTOR_DEF =
                        'OBSERVER_TARGET_VELOCITY'
     FRAME_961934_SEC_OBSERVER   = 'EARTH'
     FRAME_961934_SEC_TARGET     = -313
     FRAME_961934_SEC_ABCORR     = 'NONE'
     FRAME_961934_SEC_FRAME      = 'IAU_EARTH'
\begintext
```

Defined spacecraft-fixed frame, for pointing problems the locations and the orientations of antennas are required: SPICE system “needs to know” how antennas are oriented or rotating with respect with the reference frame fixed to the spacecraft. A comfortable way to operate with this kind of problems is to create a reference frame for each antenna. The reference frames we need to define antennas’ orientations belongs to the forth class: these antenna frames have a constant orientation with respect to the spacecraft reference frame. Hence for our scope we need to define a new reference frame (with a new name and a new frame ID) with a text kernel. Since the rotation of the antenna frame (TK frame in general) relative to the spacecraft frame (RELATIVE frame in general) is fixed (time invariant), rotation data can be provided by:

- 3x3 matrix , M, that converts vectors from the antenna frame to the spacecraft frame : $\mathbf{V}_{\mathrm{DONALD}} = M * \mathbf{V}_{\mathrm{antenna}}$;
- a set of 3 Euler angles and axes that can be used to produced M;
- a SPICE-style quaternion representing M.

The first five kernel pool variables required for TK frame specifications are the same as the Dynamic Frame defined before. For TK frames the assignments are described in Annex IV.

In order to avoid mistakes, in the following an analysis of Euler angles is done. Figure 2.9 shows Roll-Pitch-Yow convention for Euler angles.

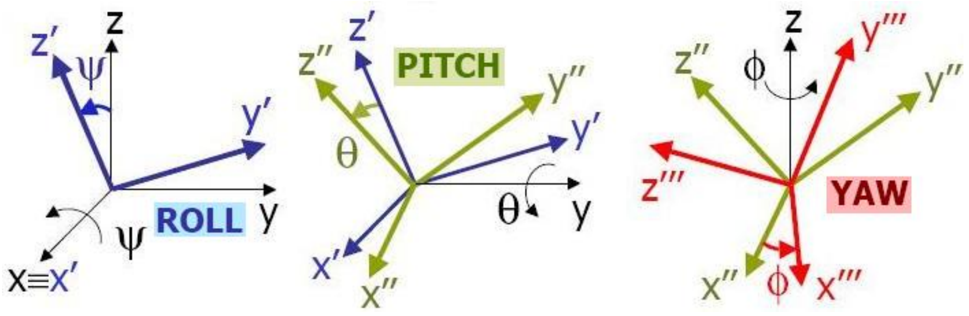


Figure 2.9: Roll-Pitch-Yow convention.

If M is the matrix that converts vectors from the RELATIVE frame to the TK frame, the angles and the axis must satisfy the relationship:

$$M = [\Phi]_{\text{axis } 3}[\theta]_{\text{axis } 2}[\psi\ ]_{\text{axis } 1}$$

where

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix} = [\psi]_1$$

$$\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} = [\theta]_2$$

$$\begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} = [\phi]_3$$

This method is particularly suitable for defining antennas reference frame: knowing how the antenna is oriented respect to the spacecraft frame through Euler angles, user can easily defined the corresponding reference frame. In this thesis we are interested to analyzed the orientation of Sun and Moon respect to the two antennas of the considered satellite (-313): the Main Antenna and the Sky Horn Antenna on board Post-EPS satellite for radiometric calibration purpose (as it will described in the next chapter). Assumed that the Main Antenna frame's name is MAIN_ANT, and that frame's matrix orientation is M=$[0°]_1[4°]_2[0°]_3$, the TK is:

```
\begindata
     FRAME_MAIN_ANT              = 26786
     FRAME_26786_NAME            = 'MAIN_ANT'
     FRAME_26786_CLASS           = 4
     FRAME_26786_CLASS_ID        = 26786
     FRAME_26786_CENTER          = -313
     TKFRAME_26786_RELATIVE      = 'DONALD'
     TKFRAME_26786_SPEC          = 'ANGLES'
     TKFRAME_26786_ANGLES        = (0,0,0)
     TKFRAME_26786_AXES          = (1,2,3)
     TKFRAME_26786_UNITS         = 'DEGREES'
\begintext
```

A similar file is required for the Sky Horn Antenna frame. Assumed that its name is SKY_HORN and its matrix orientation is M=$[0°]_1[180°]_3[4°]_2$, the TK is :

```
\begindata
     FRAME_SKY_HORN            = 16208
     FRAME_16208_NAME          = 'SKY_HORN'
     FRAME_16208_CLASS         = 4
     FRAME_16208_CLASS_ID      = 16208
     FRAME_16208_CENTER        = -313
     TKFRAME_16208_RELATIVE    = 'DONALD'
     TKFRAME_16208_SPEC        = 'ANGLES'
     TKFRAME_16208_ANGLES      = (0,180,0)
     TKFRAME_16208_AXES        = (1,2,3)
     TKFRAME_16208_UNITS       = 'DEGREES'
\begintext
```

Making an IK file

The Instrument Kernel (IK) is like a repository for instrument specific informations that may be useful in SPICE context. An IK always includes specifications for instruments' field-of-view (FOV) size, shape, orientation, beam features and so on. These kernels may also include internal instrument timing parameters and other data relating to SPICE computations and instrument geometric calibration data. Instead, instrument mounting alignment data are specified in mission's frame kernels (FK).

The main roll of IK is the definition of FOV parameters: since Post-EPS is a meteorological mission dealing with pointing problems, IK is a key file for the mission. Since IK is a SPICE text kernel, the format, the structure and the assignment rules are the typical of a text kernel. To define univocally a field of view four parameters are required:

- the shape;
- the boresight;
- the frame the boresight is defined in;

- the boundary vectors. These vector can be defined explicitly or through the half angle extents of the FOV.

Figure 2.10 shows the definition of the mentioned parameters.

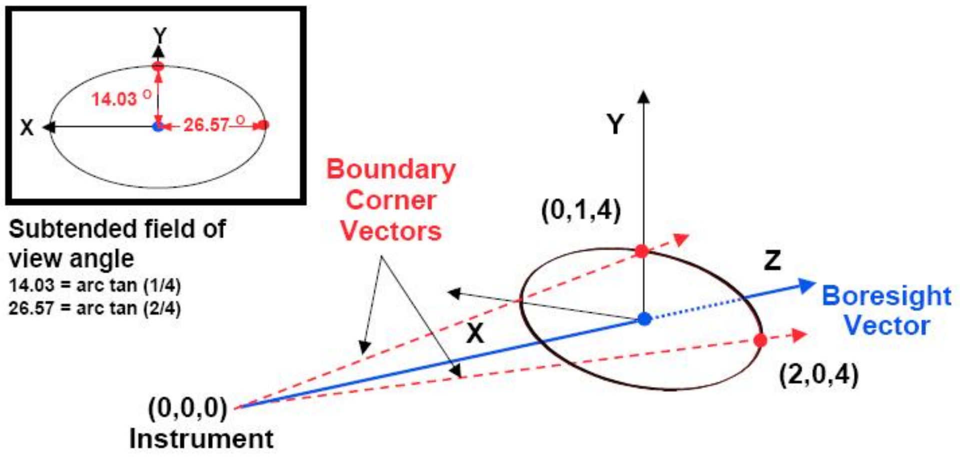


Figure 2.10: Elliptic FOV. Boundary vector can be provided either defining there components, (0,1,4) and (2,0,4), or the half angles extents, 14.3° and 26.57°.[11]

For IK frames the required assignments are reported in Annex V.

Neither the boresight nor the reference vector has to be coaligned with one of the FOV frame's axis, but for convenience, each is frequently defined to be along one of the FOV axes. Moreover neither the boresight nor corner nor reference vector has to be a unit vector, but frequently are so.

We have to create two instrument kernels, on for each antenna. Since only the Main Antenna points towards the Earth, only one IK is required to satisfy our pointing requests.

Assume that the Main Antenna (whose instrument ID is -283284) has a circular shape with a spread of three degrees. The corresponding IK is:

```
 \begindata
     INS-283284_FOV_SHAPE = 'CIRCLE'
     INS-283284_FOV_FRAME = 'MAIN_ANT'
     INS-283284_BORESIGHT = ( 0.0 0.0 1.0 )
     INS-283284_FOV_CLASS_SPEC = 'ANGLES'
     INS-283284_FOV_REF_VECTOR = ( 0.0 1.0 0.0 )
     INS-283284_FOV_REF_ANGLE = 5
     INS-283284_FOV_ANGLE_UNITS = 'DEGREES'
\begintext
```

➢ CHAPTER III: SPICE application in Post-EPS mission

This chapter is focused on the direct application of SPICE system in scenario of Post-EPS mission in order to compute satellite's position, Moon and Sun illumination angles on the antennas, and coverage.

1. Satellite position

Satellite's position is retrieved directly from the satellite's SPK that has been described in the previous chapter (paragraph 2.2). To obtain the state vectors of the satellite at a given time, *cspice_spkpos* may be used. This routine returns the state (position and velocity) of a target body relative to an observing body, optionally corrected for light time (planetary aberration) and stellar aberration. The command for the call is:

[state, lt] = cspice_spkpos (targ, et, ref, abcorr, obs)

The three components state vector is required to the SPICE routines to compute the coordinates the user needs (figure 3.1).

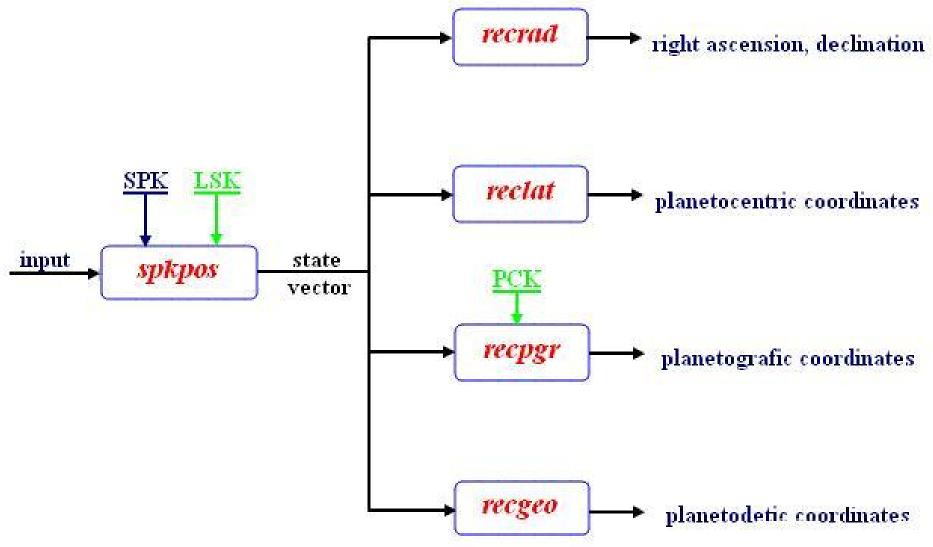


Figure 3.1: Coordinates transformations. Kernels required are: SPICE kernels (in green) provided by SPICE itself and kernels appositely created in order to conduce the analysis (in blue).

For example, in this work Post-EPS satellite's coordinates have been estimated on 11 March 2009 at 12:00:00 UTC (290044866.185526 s ET).

In order to retrieve **right ascension-declination coordinates** (figure 3.2), *cspice_recrad* is used.

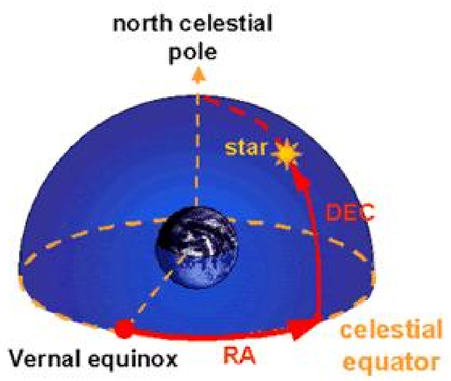


Figure 3.2: Right Ascension-Declination coordinates.[15]

The call is performed by means of the following command:

[range, ra, dec] = cspice_recrad (state)

At the given epoch satellite's coordinates are:

range = 7098.14 km ra = 189.55 deg dec = 14.33 deg

To retrieve **planetocentric coordinates** (figure 3.3), *cspice_reclat* is used.

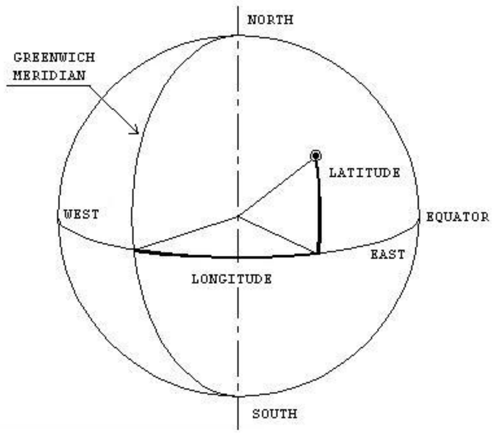


Figure 3.3: Planetocentric coordinates.

The command for the call is:

[radius, lon, lat] = cspice_reclat (state)

At the given epoch satellite's coordinates are:

range = 7098.14 km long = -170.45 deg lat = 14.33 deg

Iterating the process for a number of days of propagation, satellite's ground track can be created. The figure 3.4 shows the result obtained for Post-EPS satellite's ground track in one day of propagation.

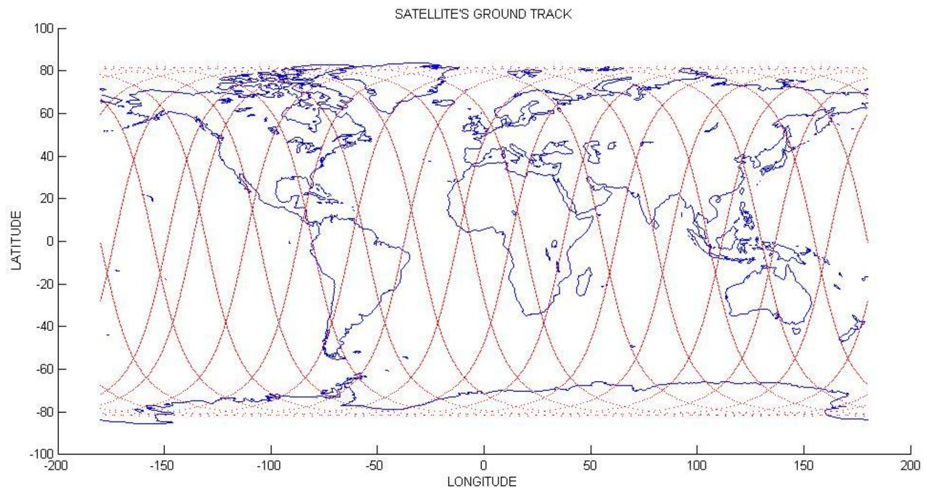


Figure 3.4: Post-EPS Satellite's ground track.

To retrieve **planetographic coordinates**, *cspice_recpgr* is used. The call is performed by means of the command:

[lon, lat, al] = cspice_recpgr(body, state, re, f)

Where ***body*** is the name of the planet with which the planetographic coordinate system is associated, ***re*** is its equatorial radius, and ***f*** is the flattening coefficient defined by the equation:

$$f = \frac{radius_{equatorial} - radius_{polar}}{radius_{equatorial}}$$

The longitude nominal range is $0 \leq$ lon $\leq 2\pi$, and the latitude nominal range is $-\pi/2 \leq$ lat $\leq \pi/2$. The figure 3.5 highlight the difference between planetocentric and planetografic coordinate systems.

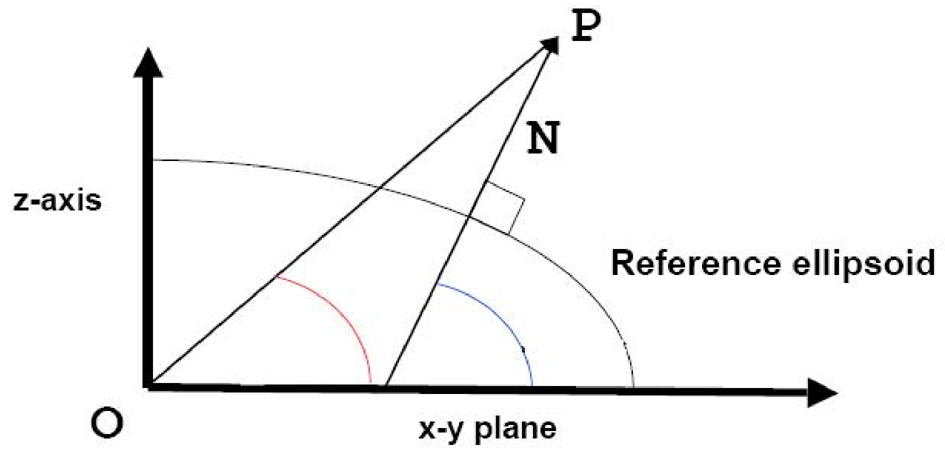


Figure 3.5: Difference between Planetocentric coordinates and Planetographic coordinates. Planetocentric latitude of a point P is the angle between segment from origin to point P and x-y plane (red arc in diagram). Planetograpfic latitude of a point P is the angle between x-y plane and extension of ellipsoid normal vector N that connects x-y plane and P (blue arc in diagram). [11]

At the given epoch satellite's coordinates are:

altitude = 721.31 km   long = 189.55 deg   lat = 14.41 deg

To retrieve **planetodetic coordinates**, *cspice_recgeo* is used. The command for the call is:

[lon, lat, alt] = cspice_recgeo(state, re, f)

Both planetografic and planetodetic coordinates refers to the reference ellipsoid but they have different nominal range.
At the given epoch satellite's coordinates are:

altitude = 721.31 km   long = -170.45 deg   lat = 14.41 deg

2. Sun and Moon Illumination Angles

Post-EPS is a meteorological observation mission. In this thesis we focus our attention on the radiometer mounted on the satellites, known as Microwave Imaging/Sounding (MWI-MWS) system. MWI-MWS is a multi-spectral microwave imager/sounding for meteorology, oceanography, sea-ice/snow/land surface observation and other climate applications.

The radiometer surveys Earth temperatures by means of measured antennas output voltages. The antenna output voltage V_a and the relative temperature T_a are linked by the equation:

$$V_a = G \cdot T_{sys} + V_{offset} = G \cdot \left(T_a + T_{rec}\right) + V_{offset}$$

where G is the global gain, T_{sys} is radiometer system temperature given by the antenna temperature T_a (which has to be measured) and the receiver temperature T_{rec}, V_{offset} is the offset voltage parameter.

Since receiver temperature and global gain are not stable due to thermal variations, it’s necessary to calibrate the radiometer with two referred temperatures (figure 3.6), a Hot Temperature and a Cold Temperature, in order to get independent the relationship from these variables.

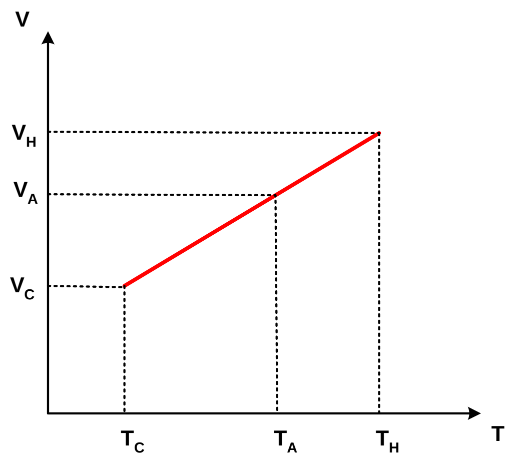


Figure 3.6: Example of Temperature calibration procedure: subscript *c* stands for *cold* and subscript *h* stands for *hot.*

Using the equation above for the calibration procedure, the following system can be written:

$$\begin{cases} V_c = G \cdot (T_c + T_{rec}) + V_{offset} \\ V_a = G \cdot (T_a + T_{rec}) + V_{offset} \\ V_h = G \cdot (T_h + T_{rec}) + V_{offset} \end{cases}$$

which allows to reduce the equation in:

$$T_a = \frac{V_a - V_c}{V_h - V_c}(T_h - T_c) + T_c$$

The hot calibration is obtained by means of a hot load (a sort of black body) observation, while the cold calibration is obtained by means of cold sky observations (where the well known 3 K of microwave background radiation are foreseen).

The radiometer system requires two antennas: an antenna (Main Antenna) pointing to the Earth collecting mission data and another antenna (so called Sky Horn Antenna) pointing to the cold sky for microwave background radiation observation.

The whole antennas system is shown in figure 3.7.

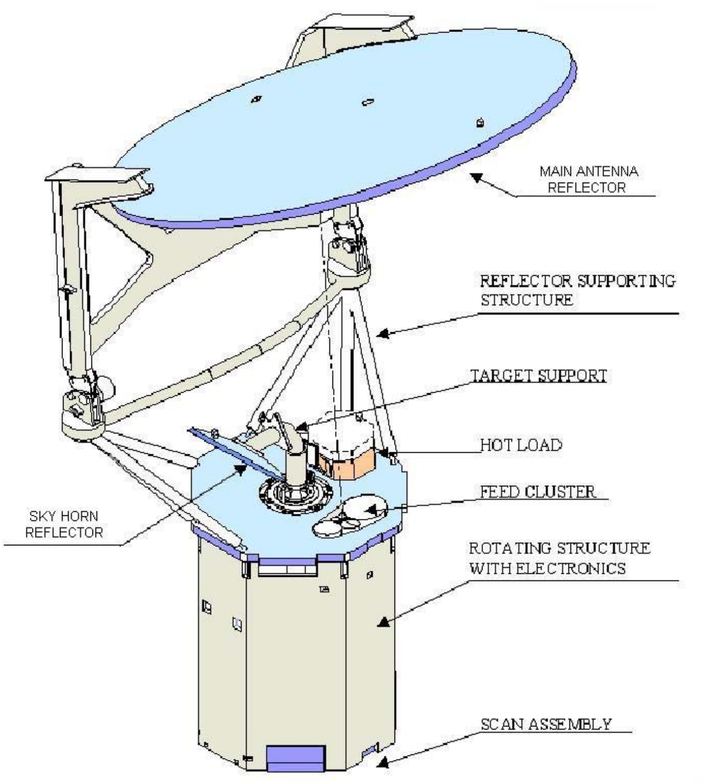


Figure 3.7: Post-EPS radiometer antennas configuration.

Temperature surveying and calibration measurements required a high degree of accuracy and they are greatly affected by Sun and Moon illumination angles respect to the measuring antennas: sunbeams from the Sun or reflected by the Moon influences antennas surveys, increasing the temperatures to measure and the noise on the data collected. Since often it is not possible to avoid that sunbeams impact on the reflectors it is extremely important to know the illumination angles of these noise sources (Sun and Moon) for each antenna of the system.

In order to retrieve Sun illumination angles (the same process has been performed for the Moon) respect to the Main Antenna (the process is the same for the Sky Horn antenna), we compute Sun position in the Main Antenna reference frame using *spkpos* SPICE routine. The Main Antenna reference frame is provided by the frame kernel created in the precedent chapter (paragraph. 2.2). The three components of the output vector are converted in azimuth-elevation coordinates using *recrad* routine. For our analysis considering the elevation complementary angle **φ** is more useful. Axes and angles convention are shown in figure 3.8.

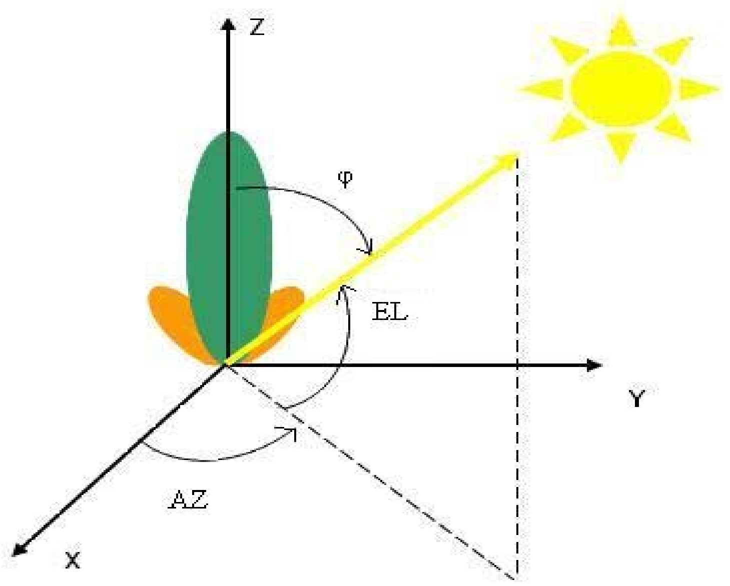


Figure 3.8: Antenna reference frame. Along the z-axis is reported the antenna pattern (main and side lobes). AZ stands for azimuth, EL stands for elevation.

In this thesis the analysis of Sun and Moon has been performed on antenna radiometer system considering a period of propagation of one year.

Figures 3.9 and 3.10 report the results obtained in this thesis for Sun and Moon illumination azimuth angle and complementary of elevation angle respect to the Main Antenna.

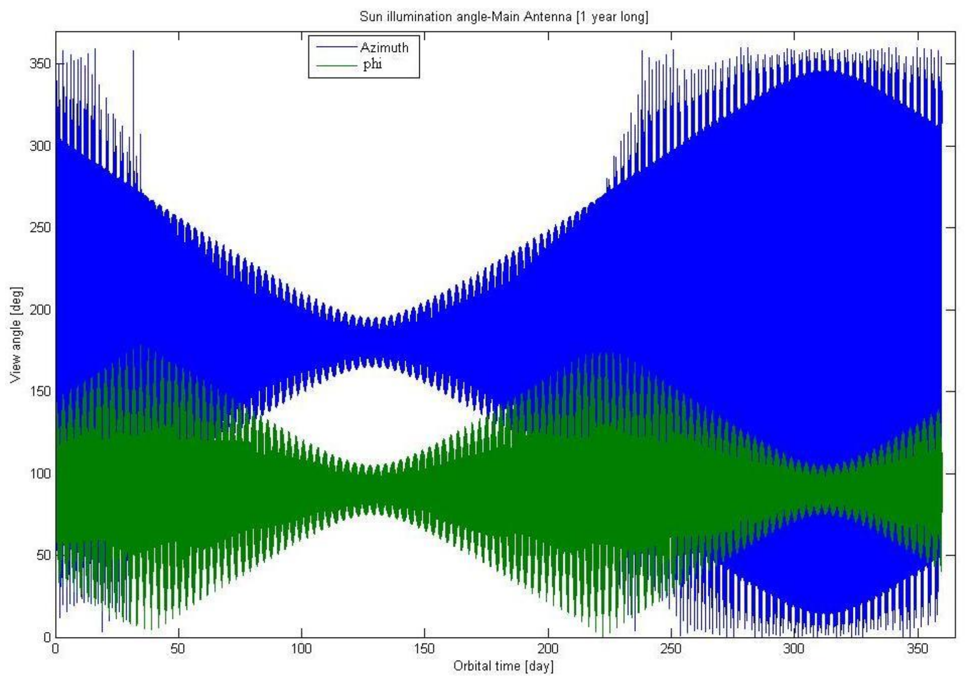


Figure 3.9: Sun illumination azimuth (in blue) and elevation complementary angle (in green) angle versus orbital time for Post-EPS radiometer Main Antenna.

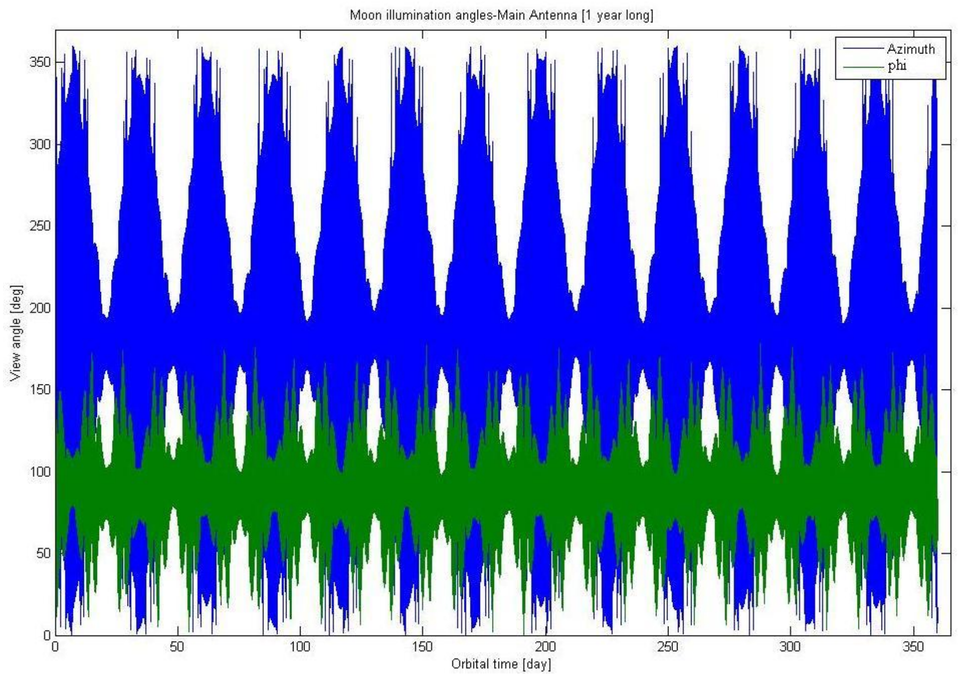


Figure 3.10: Moon illumination azimuth (in blue) and elevation complementary angle (in green) angle versus orbital time for Post-EPS radiometer Main Antenna.

In analysis of results it has to pay attention on complementary of elevation angle (**φ**): in particular for small angles, when sunbeams have a bigger intersection area with the antenna pattern, especially concerning with the main lobe. We notice that Sun unsuitable angles for the Main Antenna repeats about every 183 days, and Moon unsuitable angles about every 15 days.

Figures 3.11 and 3.12 report the results obtained in this thesis for Sun and Moon illumination azimuth and elevation angles respect to the Sky Horn Antenna.

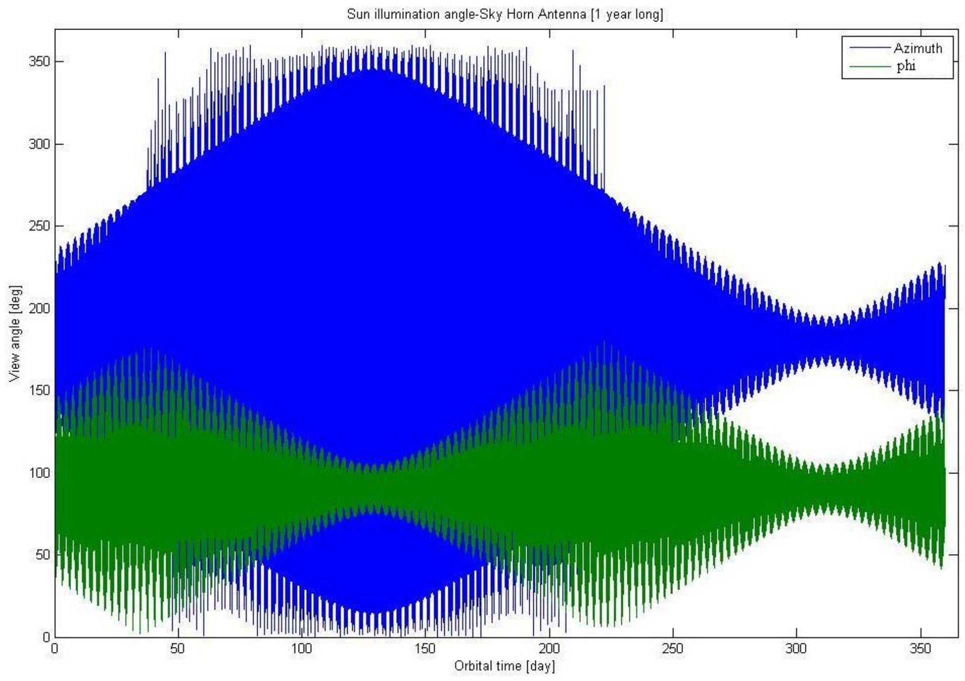


Figure 3.10: Sun illumination azimuth (in blue) and elevation complementary angle (in green) angle versus orbital time for Post-EPS radiometer Sky Horn Antenna.

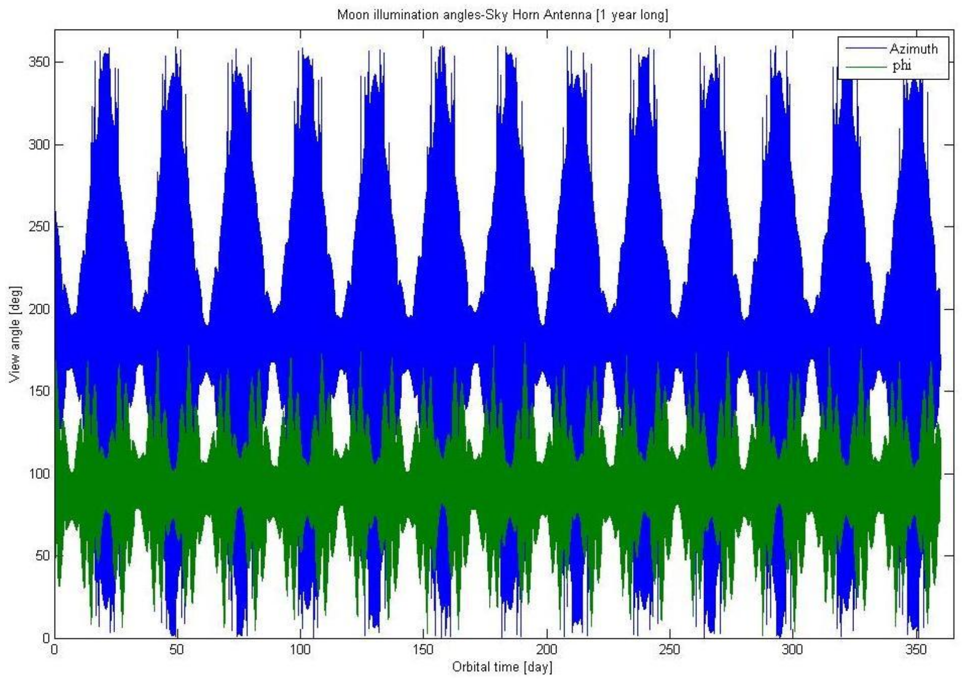


Figure 3.12: Moon illumination azimuth (in blue) and elevation complementary angle (in green) angle versus orbital time for Post-EPS radiometer Sky Horn Antenna.

Analyzing angles plot, Sun unsuitable angle for the Sky Horn repeats about every 183 days, and Moon unsuitable angles about every 15 days.

3. Coverage analysis

Coverage analysis is required in order to know in how many time the antenna sweeps the whole globe surface. Coverage mission requirements affect spacecraft's altitude, design parameters, payload final performances. Coverage analysis takes into account swath width, the area covered by the scan angle on the ground, antenna boresight orientation, FOV shape (Fig 3.13)

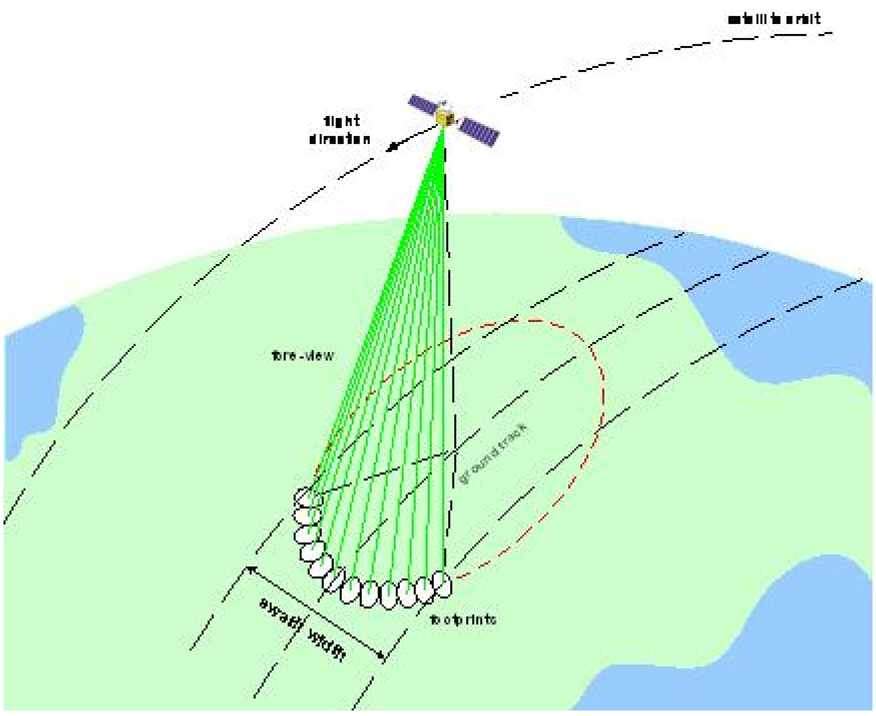


Figure 3.13:Antenna's coverage parameters.

All these informations are provided to SPICE system by an instrument kernel. Since Sky Horn Antenna performs only calibration measurements, it is not involved in pointing problems, and for this reason coverage has been analysed only for Main Antenna.

Srfxpt SPICE routines allows to compute the surface intercept point of a specified ray on a target body at a specified epoch, optionally corrected, given an observer and a direction vector defining a ray. The command for the call is:

[spoint, dist, trgepc, obspos, found]=cspice_srfxpt(method, targ, et, abcorr, obs, dref, dvec)

where ***spoint*** is the intercept point on the target***,*** ***dist*** is the distance between the observer and surface intercept***,*** ***trgepc*** is the intercept epoch***,*** ***obspos*** is the observer position in target body-fixed reference frame***,*** ***found*** is a logical flag indicating if rays intercept the surface, ***method*** provides observer surface approximation (at the moment only 'ELLIPSOID' is supported)***,*** ***dref*** is the reference frame in which *dvec* is defined***,*** ***dvec*** indicates the vectors starting from the observer. Estimated the intercept point, another routine flags on a grid map the intercept point plotting results.

Considering a swath of 1000 km, the plots in figures 3.14, 3.15 and 3.16 show coverage analysis respectively for 12, 24 and 48 hours of propagation and the related coverage ratio.

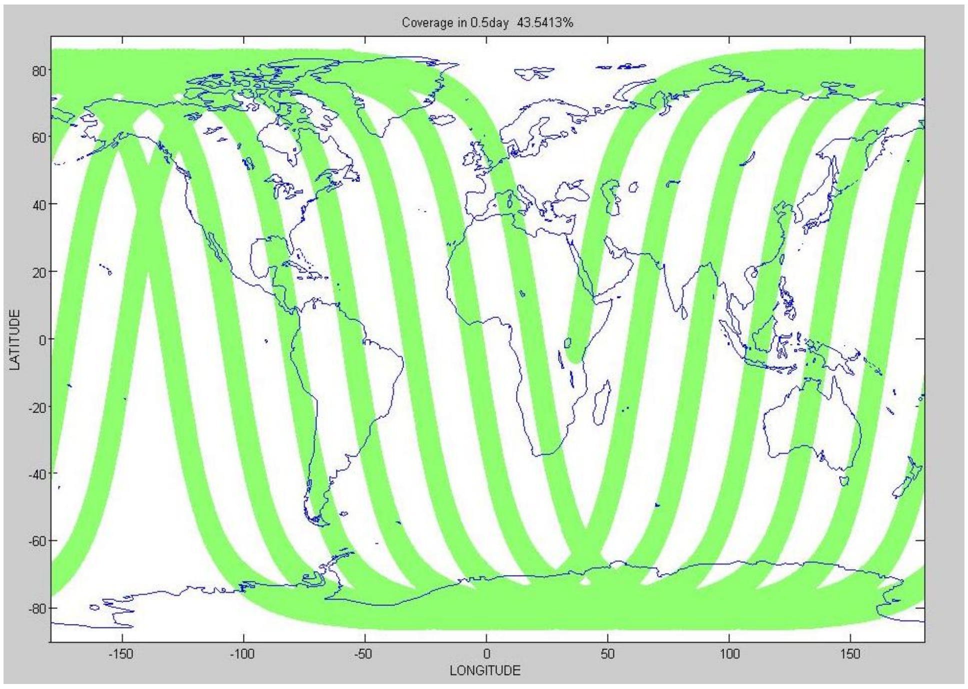


Figure 3.14: Coverage analysis and coverage ratio for Post-EPS mission in 12 h of propagation.

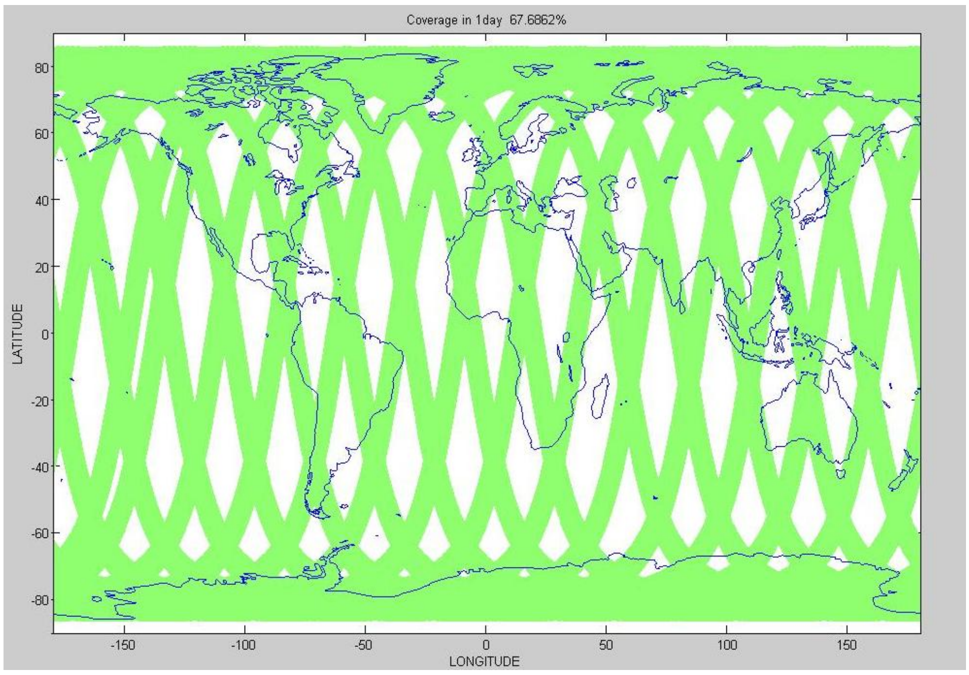


Figure 3.15: Coverage analysis and coverage ratio for Post-EPS mission in 24 h of propagation.

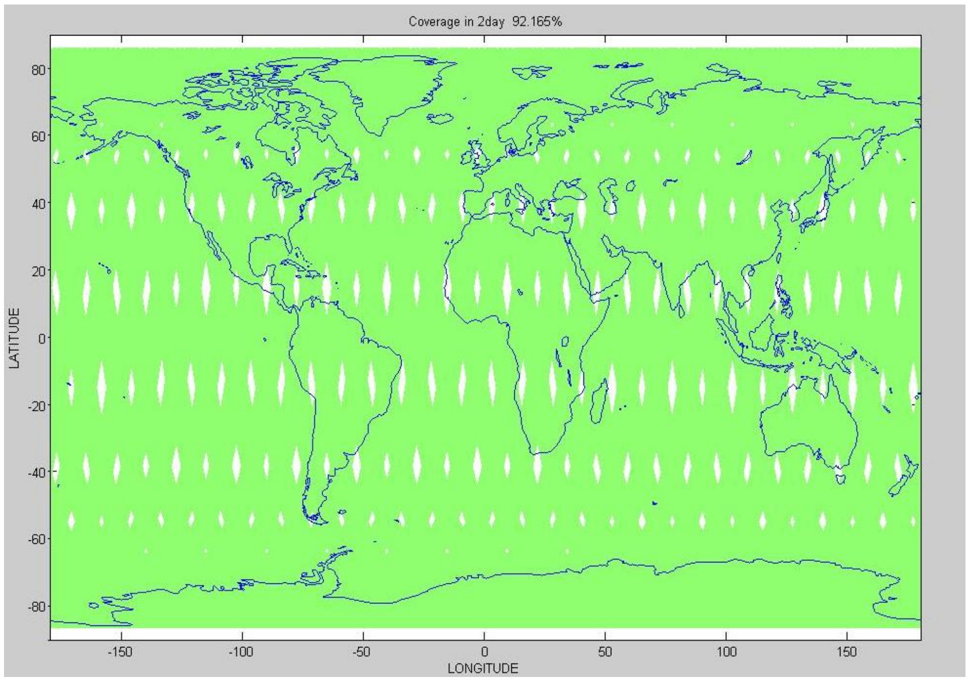


Figure 3.16: Coverage analysis and coverage ratio for Post-EPS mission in 48 h of propagation.

Referring to the figures 3.14, 3.15 and 3.16 we notes that the full coverage (more than 90%) is reached in 48 hours.

➢ ANNEX I: SPICE missions

NAIF has developed mission SPICE kernel through several years to make easier scientists' and engineers' work and to share a set of scientific informations with several partners. SPICE system dates from about 1991, but some kernels of past missions are still available. As of this writing the missions available on NAIF server are listed below.

- APOLLO. This set of data contains only a very short piece of Moon mission Apollo 15 (07/30/1971 to 08/01/1971) trajectory data in SPK format and frame kernels.
- CASSINI. This set contains all kernels regularly produced by the project of Cassini-Huygens American-European mission (1997-present) studying Saturn and its moons.
- CLEMENTINE. All type of kernels are available for the Clementine mission (02/1994 to 05/1994) officially called Deep Space Program Science Experiment (DSPSE) observing Moon and near-Earth asteroid 1620 Geographos .
- CONTOUR. This set of data provides kernels for the Comet Nucleus Tour (CONTOUR). Since the spacecraft was lost soon after the launch (07/2002), data available are the originally planned trajectory, the ephemerides for the mission's target bodies and the frame kernels.
- DAWN. This set consists of the planning and operations kernels produced for Down mission (09/2007-present) travelling forward the asteroid Vesta and the dwarf planet Ceres.

- DEEP IMPACT. This set of data provides the planning and operations SPICE kernels for the Deep Impact and EPOXI missions (1/2005-present) studying the composition of the interior of the comet 9P/Tempel by colliding a section of the spacecraft into the comet.
- DS1.This set contains all types of kernels for the Deep Space I mission (10/24/1998 to 12/18/2001) testing technologies to lower the cost and risk of future missions.
- FIDO. This set contains only a frame kernels for the FIDO Experimental Rover. These kernels were created in 1999-2000 during an unfinished attempt to implement SPICE for that vehicle.
- GLL. This set contains uncompleted kernels for Galileo mission (10/1989 to 09/2003) studying Jupiter.
- GNS. This set provides kernels regularly produced for Genesis mission (2001-present) studying solar wind.
- HAYABUSA. This set provides only SPK for HAYABUSA Japanese mission (2003-present) studying a small near-Earth asteroid named 25143 Itokawua.
- HST. This set contains only orbit data in SPK format of the Hubble Space Telescope (04/1990-present).
- IEU. This set consists of only selected trajectory data in SPK format of International Ultraviolet Explorer American-European mission (01/1978 to 07/2000) studying ultraviolet spectra.
- JUNO. This set consists of only SPK for the future mission Juno planned for 2011.
- LPM. This set provides only SPK orbit data for Lunar Prospector Mission (01/1998-07/1999).

- LUNARORBITER. Also this set contains only SPK for Lunar Orbiter mission (1966-1968).
- M01. This set contains all kernels regularly produced by the project of 2001 Mars Odyssey mission (2001-present).
- M10. This set contains a single one SPK for the mission Mariner 10 (11/1973 to 03/1975) studying mercury and Venus. The available kernel covers only from 03/24/1974 to 04/04/1974, which is the period of the first Mercury flyby.
- M9. This set contains a single SPK kernel and an SCLK kernel for the mission Mariner 9 (05/1971 to 10/1972).
- MCO. This set provides all kernels produced by the project Mars Climate Orbiter (12/1998 to 10/1999) during the cruise to Mars before being destroyed.
- MER. This set contains kernels regularly produced for the mission Mars Exploration Rovers (2003-present).
- MESSENGER. This set contains all kernels for Messenger mission (08/2004 -present) studying Mercury.
- MEX. This set contains kernels regularly produced for the European mission Mars Express (06/2003-present) .
- MGN. This set provides only SPK kernels produced for Magellan mission (08/1990 to 08/1994) studying Venus.
- MGS. This set contains kernels for the mission Mars Global Surveyor (11/1996 to 11/2006).
- MPF. This set contains all kernels archived for the mission Mars Pathfinder (12/1996 to 03 1998)
- MPL. This set provides SPK kernels produced for the mission Mars Polar Lander (01/1999-12/1999). Also kernels produced in

anticipation of a successful landing (which didn't occur since the spacecraft was destroyed) are available.

- MRO. This set provides kernels for the mission Mars Reconnaissance Orbiter (03/2006-present).
- MSL. This set consists only of SPK and FK kernels produced for the future mission Mars Science Laboratory planned for 2011.
- NEAR. This set contains kernels produced for Near Earth Asteroids (02/1996 to 02/2001).
- NOZOMI. This set provides only trajectory data for Nozomi Japanese mission (02/1999 to 01/2004). Data coverage is from the launch to the slightly flyby of Mars due to inability to perform orbit insertion in Mars.
- PHOBOS. This set consists only of two trajectory data in SPK format produced for the Russian mission Phobos 88 (01/1989 to 03/1989), before contacts with the spacecraft was permanently lost during Mars orbit.
- PHOENIX. This set provides all kernels produced for PHOENIX mission studying Mars (08/2007-present).
- PIONEER 10. This set contains a single SPK file for the spacecraft, Jupiter, the Galilean satellites, Earth and Sun produced for the mission Pioneer 10 (03/1972-undefined) studying deep space. It's a merge of several SPK. Data are provided from the launch to 01/1990.
- PIONEER 11. This set contains a single SPK file for the spacecraft, Jupiter, the Galilean satellites, Earth and Sun produced for the mission Pioneer 11 (04/1973-undefined) studying deep space. It's a merge of several SPK. Data are provided from the launch to 01/1990.

- PIONEER 6. This set contains a single SPK file for the spacecraft, the planets, the Earth and the Sun produced for the mission Pioneer 6 (12/1965-undefined) studying deep space. It's a merge of several SPK. Data are provided from 01/1996 to 12/1999.
- PIONEER 8. This set contains a single SPK file for the spacecraftand the Earth produced for the mission Pioneer 8 (12/1967-undefined) studying deep space. It's a merge of several SPK. Data are provided from 07/1997 to 12/1999.
- ROCKY 7. This set contains kernels for Rocky 7 Experimental Rover field test (05/1997). These kernels were created to demonstrate the applicability of SPICE for a rover mission.
- ROSETTA. This set provides all kernels for Rosetta European mission (03/2004-present), studying the comet 67P/Churyumov-Gerasimenko.
- SDU. This set provides all kernels regularly produced for Stardust mission (02/1999-01/2006) investigating the makeup of the comet Wild 2 and its coma.
- SELENE. This set will contain kernels produced for Selene Japanese mission (09/2007-present).
- SIRTF. This set provides SPK and SCLK kernels produced for the Space Infrared Telescope Facility, SIRTF, mission (08/2003-present).
- SMART1. This set contains all kernels produced for Small Missions for Advanced Research in Technology, SMART, European mission (09/2003 to 09/2006).
- ULYSSES. This set contains only trajectory data in SPK format produced for the Ulysses American-European mission (10/1990 to 06/2008) studying the Sun at all latitudes.

- VEGA. This set provides a single SPK for the Russian Vega1 (12/1984-present) and Vega 2 (12/1984-present) spacecraft and the comet Halley during their flyby. Data are provided from 03/01/1986 to 03/17/1986.
- VEX. This set provides all kernels for Venus Express European mission (10/2005-present).
- VIKING. This set provides all kernels for the two Viking orbiters and two SPK for the landed location of the Viking Landers (1975 to 1982), studying Mars.
- VOYAGER. This set provides a volatile and eclectic collection of kernels made from assorted data produced by Voyager missions (1977-undefined) studying deep space. Data are provided from 07/1997 to 2050.

➢ ANNEX II: Setup File Generation

The required assignments in a setup file are:

`INPUT_DATA_FILE`	=	`'`input ephemeris data file name`'`
`OUTPUT_SPK_FILE`	=	`'`output SPK file name`'`
`INPUT_DATA_TYPE`	=	`'STATES'` or `'ELEMENTS'` or `'EQ_ELEMENTS'` or `'TL_ELEMENTS'`
`OUTPUT_SPK_TYPE`	=	5 or 8 or 9 or 10 or 12 or 13 or 15 or 17
`OBJECT_ID`	=	numeric code assigned to the object. Either `OBJECT_ID` or `OBJECT_NAME` is to be used. If this assignment is absent if this `OBJECT_NAME` is required. If NAIF has not assigned an ID code the user may select a temporary ID. This ID must be a negative number for a spacecraft.
`OBJECT_NAME`	=	`'`NAIF supported object name`'` The name has to be NAIF supported for the object. This keyword is required if the `OBJECT_ID` keyword is absent. If

		both keywords are present, the MKSPK program uses the `OBJECT_ID` and ignores this assignment.
`CENTER_ID`	=	numeric code assigned to the body which is the center of the motion for the object. If this keyword is absent, the `CENTER_NAME` keyword must be used.
`CENTER_NAME`	=	`'`NAIF supported body name`'` This is one if the NAIF supported names for the center of motion. This keyword is required if the `CENTER_ID` keyword is absent. If both keywords are present the MKSPK program uses the `CENTER_ID` and ignores this assignment.
`REF_FRAME_NAME`	=	`'`reference frame name`'` If this is not a standard frame supported by NAIF, the definition for this frame must be present in a frame definition kernel file specified in the `FRAME_DEF_FILE` keyword.

`PRODUCER_ID`	=	'producer identifier'
`DATA_ORDER`	=	'ordered list of input parameter names' The names must be delimited with white Space.
`DATA_DELIMITER`	=	'delimiter separating input data items' User can select : `'TAB'`, `'EOL'`, comma (`','`), semicolon (`';'`) or white space (`' '`). Only one of these values can be used.
`LEAPSECONDS_FILE`	=	'leapseconds file name'

More assignments may be needed depending on the data type and other conditions. The conditional assignments are:

`PCK_FILE`	=	('PCK_1 file name' 'PCK_2 file name' ... 'PCK_n file name') This keyword may be absent in the case when you do not need PCK constants or when these constants are defined with other setup keywords.
`FRAME_DEF_FILE`	=	'frame definition file name'

```
                                   This keyword is required
                                   REF_FRAME_NAME keyword is absent. In
                                   this case the frame must be defined in a
                                   frames definition kernel file.
COMMENT_FILE       = 'comment file name'
INPUT_DATA_UNITS   = ( 'ANGLES =  angle unit'
                       'DISTANCES=  distance unit'  )
                     MKSPK recognizes: RADIANS,
                     DEGREES, ARCMINUTES,
                     ARCSECONDS, HOURANGLE,
                     MINUTEANGLE, SECONDANGLE,
                     METERS, KM, CM, MM, FEET,
                     INCHES, YARDS, STATUTE_MILES,
                     NAUTICAL_MILES, AU, PARSECS,
                     LIGHTSECS, LIGHTYEARS. Note that
                     MKSPK assumes that the time units of
                     any input data parameters or constants
                     specified in the setup file are seconds.
EPOCH_STR_LENGTH   = length of epoch string.
                     This length must be provided to the
                     program in order to enable parsing of the
```

		input data records containing two epoch strings.
`IGNORE_FIRST_LINE`	=	number of initial lines to be ignored while reading input file
`LINES_PER_RECORD`	=	number of lines in one input record
`TIME_WRAPPER`	=	'# time wrapper' This assignment is the specification of alphanumeric characters that are used to define the "`EPOCH`" strings. As example, `'# TDT'` or `'# ETSECONDS'`.
`START_TIME`	=	'start time' If this value is absent the MKSPK program calculates it using data
`STOP_TIME`	=	'stop time' If this value is absent the MKSPK program calculates it using data
`PRECESSION_TYPE`	=	`'NO PRECESSION'` or `'APSIDE PRECESSION ONLY'` or `'NODE PRECESSION ONLY'` or `'APSIDE AND NODE PRECESSION'`
`POLYNOM_DEGREE`	=	polynomial degree of Lagrange or

		Hermite interpolation
`CENTER_GM`	=	center GM value. If it is absent, the MKSPK program attempts to find this value in a PCK file.
`CENTER_POLE_RA`	=	the right ascension of the center's north pole given with respect to the reference frame.
`CENTER_POLE_DEC`	=	the declination of the center's north pole given with respect to the reference frame.
`CENTER_J2`	=	center's J_2 value. If it is absent, the MKSPK program attempts to find this value in a PCK file.
`CENTER_EQ_RADIUS`	=	center's equatorial radius. If it is absent, the MKSPK program attempts to find this value in a PCK file.
`SEGMENT_ID`	=	`'`segment identifier`'`
`APPEND_TO_OUTPUT`	=	flag indicating whether new segments should or shouldn't be appended to an existing SPK file. This keyword can be `'YES'` or `'NO'` .

➢ ANNEX III: Orbital Propagator for Post EPS mission

```
%************************************************************
%********  Orbital propagator for Post-EPS mission  ********
%************************************************************
%********************* Start of Header *********************
% PROGRAM NAME:              propag_SSO.m
% DATE:                      09 March 2009
% VERSION:                   1.0
% PURPOSE:                   This program computes
%                            satellite’s coordinates and
%                            velocities in ECEF frame and
%                            create an ephemeris text file.
%
% INPUT ARGUMENTS:           Orbital parameters, initial
%                            propagation date,
%                            propagation days and
%                            propagation step time.
%
% OUPUT ARGUMENTS:           ephemeris text file
%
% AUXILIARY SUBROUTINES:    SPICE routines
%********************** End of Header **********************
%************************************************************


%service operation

    clc
    clear all
    close all
    current_directory=pwd;
    addpath(strcat(current_directory,'\mice\src\mice'));
    addpath(strcat(current_directory,'\mice\lib'));
    cspice_kclear

% Universal, Earth and orbital constants

    %Earth rotation velocity [rad/s]
    omegaearth=7.2921158553e-5;
    %gravitational parameter [Km^3/s^2]
    mi=398600.4415 ;
    J2=0.0010826269;
    %mean Earth equatorial radius [km]
    Req=6378.13649;
    %Sun precession [deg/day]
    alfapuntosun=0.985647332;

%orbital parameters and initial values

    %height [Km]
    H  = 720;
    %satellite orbital radius [km]
    a=Req+H;
    %propagation days
    Nd = 10;
    %time step [sec]
    dt = 10;
    %initial propagation date
    str= '12 FEB 2009 12:00:00';
    %initial true anomaly [rad]
    niC(1)=0;
    %longitude of the ascending node [rad]
    OMC=37.5*cspice_rpd;
    %argument of the periapsis [rad]
    omC=90*cspice_rpd;
    %eccentricity 0.00118;
    eC=0;
    %major semi-axis [km]
    aC=a;
    %minor semi-axis [km]
    bC=aC*sqrt(1-eC^2);
    %mean motion [rad/sec]
    n=sqrt(mi/(a^3));
    % mean orbital period [s];
    T = 2*pi/n;
    %Keplerian period [s]
    perkeps=2*pi/n;

% compute orbital inclination

    %Sun precession convertion in [rad/sec]
    alfapuntosun=(alfapuntosun*(pi/180))/(3600*24);
    inc=acos(alfapuntosun/(-3/2*J2*sqrt(mi/a^3)*(Req/a)^2));

%time vector creation

    t = 0:dt:Nd*86400;
    npunti=length(t);

%time conversion using SPICE routines

    cspice_furnsh= …
    strcat(current_directory,'\mice\lib\naif0009.tls');
    et = cspice_str2et( str );

%Initial values for eccentric and mean anomaly

    sinEC0=sqrt(1-eC^2)*sin(niC(1))/(1+eC*cos(niC(1)));
    cosEC0=(eC+cos(niC(1)))/(1+eC*cos(niC(1)));
    EC(1)=atan2(sinEC0,cosEC0);
    MC(1)=EC(1)-eC*sin(EC(1));

%compute mean and eccentric anomaly

    for it=2:npunti
        MC(it)=MC(it-1)+n*dt;
        %Newton beginning
        dEC=1;
        iter=1;
        ECtry(iter)=MC(it);
    while dEC>10^-6 && iter<10
        iter=iter+1;
        ECtry(iter)=ECtry(iter-1)-(ECtry(iter-1)-eC*sin
        (ECtry(iter-1))-MC(it))/(1-eC*cos(ECtry(iter-1)));
        dEC=(ECtry(iter)-ECtry(iter-1))/ECtry(iter-1);
    end
    EC(it)=ECtry(iter);
    sinniC=sin(EC(it))*sqrt(1-eC^2)/(1-eC*cos(EC(it)));
    cosniC=(cos(EC(it))-eC)/(1-eC*cos(EC(it)));
    niC(it)=atan2(sinniC,cosniC);
    time(it)=(it-1)*dt;
    end

 %satellite ECI position [km]

    radiusC=aC*(1-eC*cos(EC));
    satposX=radiusC.*(cos(omC+niC).*cos(OMC)-sin(omC+niC).
    *sin(OMC).*cos(inc));
    satposY=radiusC.*(cos(omC+niC).*sin(OMC)+sin(omC+niC).
    *cos(OMC)*cos(inc));
    satposZ=radiusC.*sin(omC+niC).*sin(inc);

%Coefficients for velocity computations

    l_1=cos(OMC)*cos(omC)-sin(OMC)*sin(omC)*cos(inc);
    m_1=sin(OMC)*cos(omC)+cos(OMC)*sin(omC)*cos(inc);
    n_1=sin(omC)*sin(inc);
    l_2=-cos(OMC)*sin(omC)-sin(OMC)*cos(omC)*cos(inc);
    m_2=-sin(OMC)*sin(omC)+cos(OMC)*cos(omC)*cos(inc);
    n_2=cos(omC)*sin(inc);

%Satellite ECI velocity [km/s]

    satvelX=n*aC./radiusC.*(bC.*l_2.*cos(EC)-aC.*l_1.
    *sin(EC));
    satvelY=n*aC./radiusC.*(bC.*m_2.*cos(EC)-aC.*m_1.
    *sin(EC));
    satvelZ=n*aC./radiusC.*(bC.*n_2.*cos(EC)-aC.*n_1.
    *sin(EC));

%conversion from ECI to ECEF
```

```
    %rotation velocity of ECEF with respect to ECI
    w=[0;0;omegaearth];
    to=t(1);
    for it=1:npunti
        t_k=t(it);
        theta(it)=omegaearth*(t_k-to);
        %transformation matrix
        R(1,:)=[cos(theta(it)),sin(theta(it)),0];
        R(2,:)=[-sin(theta(it)),cos(theta(it)),0];
        R(3,:)=[0,0,1];
        satpos=[satposX(it),satposY(it),satposZ(it)];
        satvel=[satvelX(it),satvelY(it),satvelZ(it)];
        satposECEF(:,it)=R*satpos';
        satvelECEF(:,it)=R*satvel';
    end

%print ephemeredes in output file

    eph= fopen('ephemerisdata.txt','wt');
    EPHEMERIS = [time+et;satposECEF;satvelECEF ];
    for ii = 1:npunti
        for kk=1:7
            fprintf(eph,'%12.6f\t',EPHEMERIS(kk,ii));
            end
    fprintf(eph,'\n');
    end
    fclose (eph);
    type ephemerisdata.txt
```

➢ ANNEX IV: Frame Kernel Generation

The required assignments in a two vectors frame kernel are:

`FRAME_<f_name>`	= `<f_ID>` The numeric code assigned to the reference frame.
`FRAME_<f_ID>_NAME`	= `<f_name>` The name chosen for the frame must not exceed 26 characters.
`FRAME_<f_ID>_CLASS`	= The numeric code that identifies the class frame
`FRAME_<f_ID>_CLASS_ID`	= `<f_ID>`
`FRAME_<f_ID>_CENTER`	= `<spacecraft_ID>` The numeric code for the object chosen as the center of the new reference frame. This

```
                                           assignment allows to
                                           connect an object to his
                                           body-fixed frame.
FRAME_<f_ID>_RELATIVE             = 'Reference frame name'
                                    The reference frame
                                    name the new frame is
                                    oriented with.
FRAME_<f_ID>_DEF_STYLE            = 'PARAMETERIZED'
FRAME_<f_ID>_FAMILY               = 'TWO-VECTOR'
            Or 'MEAN_EQUATOR_AND_EQUINOX_OF_DATE'
            Or 'TRUE_EQUATOR_AND_EQUINOX_OF_DATE'
            Or 'MEAN_ECLIPTIC_AND_EQUINOX_OF_DATE'
            Or 'EULER'
FRAME_<f_ID>_PRI_AXIS             = 'First axis'
FRAME_<f_ID>_PRI_VECTOR_DEF       = 'Primary vector'
                        Suitable chances are:
                        'OBSERVER_TARGET_POSITION'
                        'OBSERVER_TARGET_VELOCITY'
                        'TARGET_NEAR_POINT'
                        'CONSTANT'
FRAME_<f_ID>_PRI_FRAME            = 'Reference frame name'
```

```
                                            Reference frame name
                                            the primary vector is
                                            defined in.
FRAME_<f_ID>_PRI_OBSERVER       = 'Observer's name'
FRAME_<f_ID>_PRI_TARGET         = 'Target's name'
FRAME_<f_ID>_PRI_ABCORR         = 'Flag for corrections'
FRAME_<f_ID>_SEC_AXIS           = 'Second axis'
FRAME_<f_ID>_SEC_VECTOR_DEF     = 'Secondary vector'
                             Suitable chances are:
                                'OBSERVER_TARGET_POSITION'
                                'OBSERVER_TARGET_VELOCITY'
                                'TARGET_NEAR_POINT'
                                'CONSTANT'
FRAME_<f_ID>_SEC_FRAME          = 'Reference frame name'
                                  Reference frame name
                                  the secondary vector is
                                  defined in.

FRAME_<f_ID>_SEC_OBSERVER       = 'Observer's name'
FRAME_<f_ID>_SEC_TARGET         = 'Target's name'
FRAME_<f_ID>_SEC_ABCORR         = 'Flag for corrections'
```

For TK frames the assignments are:

```
FRAME_<f_name>                 = <f_ID>
FRAME_<f_ID>_NAME              = <f_name>
FRAME_<f_ID>_CLASS             = class ID
FRAME_<f_ID>_CLASS_ID          = <f_ID>
FRAME_<f_ID>_CENTER            = <spacecraft_ID>
TKFRAME_<f_ID>_RELATIVE        = 'Relative  frame name'
TKFRAME_<f_ID>_SPEC            = 'MATRIX'
                                 or 'ANGLES'
                                 or 'QUATERNION'
```

According to the choice, in the following each specification is described.

To define a rotation using a matrix, the assignments are:

```
TKFRAME_<f_ID>_SPEC            = 'MATRIX'
TFRAME_<f_ID>_MATRIX           = (  matrix_value [1][1],
                                    matrix_value [2][1],
                                    matrix_value [3][1],
                                    matrix_value [1][2],
                                    matrix_value [2][2],
                                    matrix_value [3][2],
                                    matrix_value [1][3],
                                    matrix_value [2][3],
                                    matrix_value [3][3] )
```

To define a rotation using a Euler angles (positive counterclockwise), the assignments are:

```
TKFRAME_<f_ID>_SPEC            = 'ANGLES'
TKFRAME_<f_ID>_ANGLES          = (angle1, angle2, angle3)
TKFRAME_<f_ID>_AXES            = (axis 1,  axis 2,  axis 3)
                                 The axes must be chosen
                                 from the set of integers
                                 1,2,3 where 1 stands for
                                 the x-axis, 2 for the y-
                                 axis, and 3 for the z-axis.
TKFRAME_<f_ID>_UNITS           = 'angle units'
```

➢ ANNEX V: Instrument Kernel Generation

The required assignments for instrument kernels are:

`INS<in_ID>_FOV_SHAPE`	= `'CIRCLE'` or `'ELLIPSE'` or `'RECTANGLE'` or `'POLYGON'` The instrument ID must be a negative number.
`INS<in_ID>_FOV_FRAME`	= 'Reference frame' The name of the reference frame the boresight and the boundary vectors are defined in.
`INS<in_ID>_BORESIGHT`	= `( X, Y, Z )`

In the case of explicit boundary vectors definition:

```
INS<in_ID>_FOV_CLASS_SPEC      = 'CORNERS'
INS<in_ID>_FOV_BOUNDARY_CORNERS = ( X1, Y2, Z3,
                                   … … …
                                   Xn, Yn, Zn )
```

In the case of half angles extents definition:

```
INS<instr_ID>_FOV_CLASS_SPEC  = 'ANGLES'

INS<instr_ID>_FOV_REF_VECTOR  = ( X, Y, Z )
```

Reference vector that, together with the boresight vector, define the plane in which the half angle is measured.

```
INS<instr_ID>_FOV_REF_ANGLE   = halfangle1
INS<instr_ID>_FOV_CROSS_ANGLE = halfangle2
```

This angle is measured in the plane normal to last plane and containing the boresight vector.

```
INS<instr_ID>_FOV_ANGLE_UNITS = 'angle units'
```

➢ CONCLUSIONS

In this thesis an analysis of source noise and coverage of the radiometer which will be installed on board of MetOp satellite for Post-EPS mission has been performed by means of a tool developed in MATLAB software language. The tool takes great advantage of SPICE routines creating a powerful and useful source for geometrical and radiometric analysis. SPICE is a library, provided by the NAIF node of NASA Planetary Data System, which allows scientists and engineers to share data analysis and scientific results come from past mission to improve future mission analysis.

This study has pointed out how powerful and high level performances SPICE system is. Creating apposite kernels and using SPICE “built-in” routines we have developed a complete software that with few basic inputs, such as orbital and propagation parameters, retrieve geometrical and instrumental data fundamental for mission analysis. Figure C.1 summarises software structure showing required inputs and the logical steps developed and automated to retrieve needed results. Kernels required are: SPICE kernels (in green) provided by SPICE itself and kernels appositely created in order to conduce the analysis (in blue). Moreover, all routines enclosed in blue rectangles are already provided by SPICE system, while routines in yellow rectangles are not provided by NAIF.

This thesis has been focused on the computation and analysis of Sun and Moon illumination angles on radiometer’s antennas and instrument coverage. These analyses are very important for mission success: sunbeams come from the Sun or reflected by the Moon influences antennas surveys, increasing the temperatures to measure and the noise on the data collected. Since often it is not possible to avoid that sunbeams impact on the reflectors it is extremely important to know the illumination angles of these noise sources (Sun and Moon) for each antenna of the system. Also coverage analysis is fundamental: coverage mission requirements affect spacecraft's altitude, design parameters and payload final performances. As reported in chapter 2, for Post-EPS mission the full coverage (more than 90%) is reached in 48 hours.

Regarding to the developed software, it has the advantage of remaining valid also for different mission frameworks employing pointing problems, changing instrument and mission parameters. Of course the implementation in the future of a more detailed orbital propagator will allow to analyze better mission performances and consequently to get better instrumental analysis.

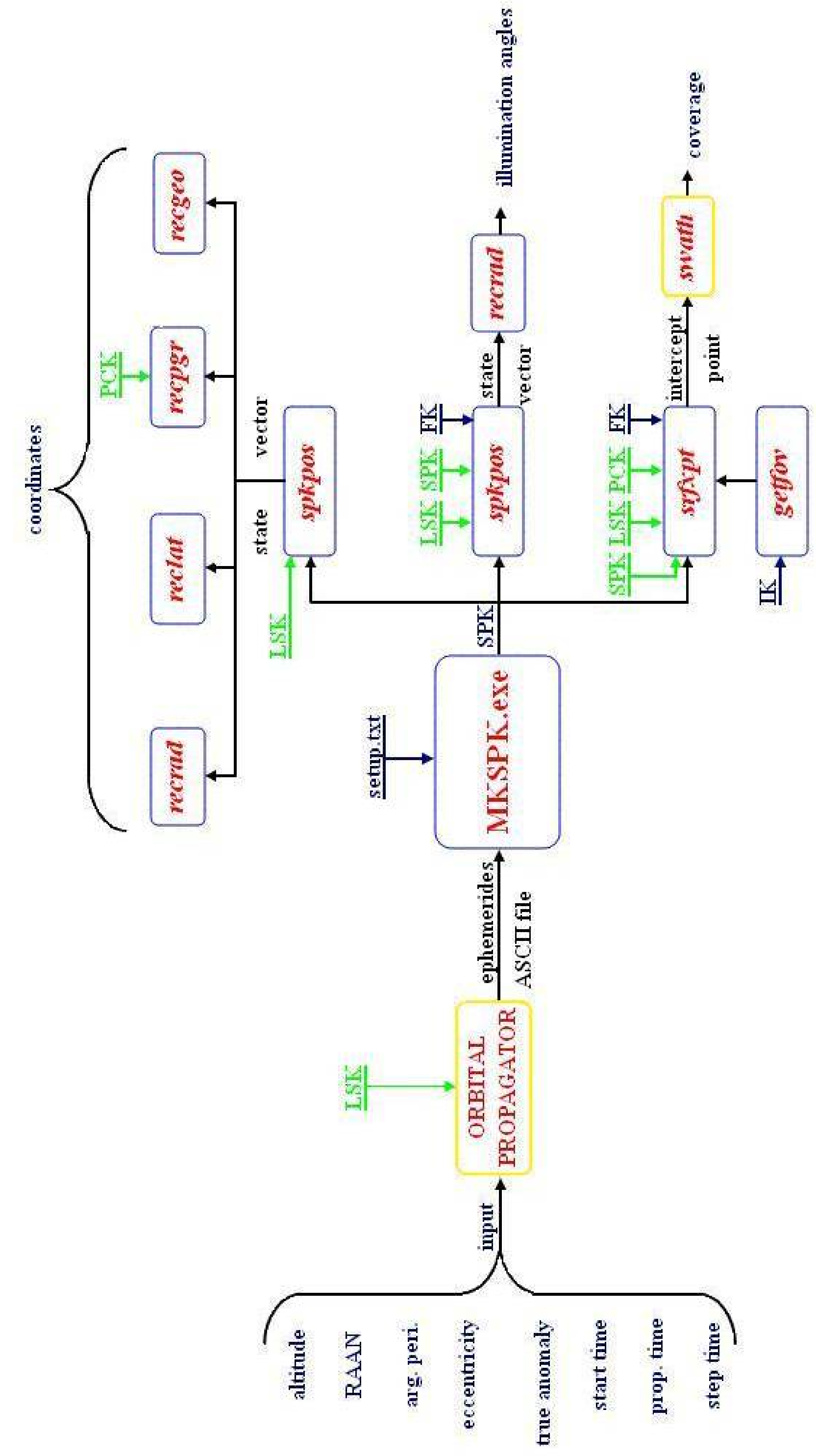


Figure C1: logical summary of the developed software.

➢ ACRONYMS

API: *Application Programming Interface*

ASCII: *American Standard Code for International Interchange*

AU: *Astronomical Unit*

AZ: *AZimuth*

CK: *Camera Kernel*

CONTOUR: *COmet Nucleus TOUR*

DEC: *Declination*

DS: *Deep Space*

DSPSE: *Deep Space Program Science Experiment*

ECEF: *Earth Centered Earth Fixed*

ECI: *Earth Centered Inertial*

EK: *Event Kernel*

EL: *ELevation*

EPS: *EUMETSAT Polar System*

ET: *Ephemeris Time*

EUMETSAT: *European Organisation for the Exploration of Meteorological Satellites*

FES2004: *Finite Element Solution tidal atlates 2004*

FK: *Frame Kernel*

FOV: *Field Of View*

GGM02: *GRACE Gravity Model 2002*

GLL: *Galileo*

GNS: *Genesis*

GOS: *Global Operational Satellite Observation System*

HST: *Hubble Space Telescope*

IAU: *International Astronomical Union*

ICFR: *International Celestial Reference Frame*

IEU: *International Ultraviolet Explorer*

IJPS: *Initial Joint Polar-Orbiting Operational Satellite System*

IK: *Instrument Kernel*

LEO: *Low Earth Orbit*

LLR: *Lunar Laser Range*

LPM: *Lunar Prospector Mission*

LTDN: *Local Time of Descending Node*

MCO: *Mars Climate Orbiter*

MDA: *Modified Difference Arrays*

MER: *Mars Exploration Rover*

MetOP: *Meteorological Operational*

MEX: *Mars Express*

MEX: *Matlab External Interface*

MGN: *Magellan*

MGS: *Mars Global Surveyor*

MO: *Mars Odyssey*

MPF: *Mars PathFinder*

MPL: *Mars Polar Lander*

MRO: *Mars Reconnaissance Orbiter*

MSL: *Mars Science Laboratory*

MWI: *MicroWave Imaging*

MWS: *MicroWave Sounding*

NAIF : *Navigation and Ancillary Information Facilities.*

NOAA: *National Oceanic an Atmospheric Administration*

NWP: *Numerical Weather Prediction*

PDS: *Planetary Data System.*

PCK: *Planet Constant Kernel*

PPN: *Parameterized Post Newtonian formalism*

RA: *Right Ascension*

SCL: *Spacecraft Clock*

SIRTF: *Space InfraRed Telescope Facility*

SMART: *Small Missions for Advanced Research in Technology*

SPICE: *Spacecraft Planet Instrument C-matrix Events*

SPK: *Spacecraft Kernel*

SSB:	*Solar System Barycentre*
SSO:	*Sun-Synchronous Orbit*
TAI:	*International Atomic Time*
TDB:	*Barycentric Dynamical Time*
TDT:	*Terrestrial Dynamical Time*
TK:	*Text Kernel*
UT:	*Universal Time*
UTC:	*Coordinated Universal Time*
VEX:	*Venus EXpress*

➢ BIBLIOGRAPHY

[1]: C. Acton, N. Bachman, L. Elson, B. Semenov, F. Turner, E. Wright,

Caltech/Jet Propulsion Laboratory *Extending NASA's SPICE Ancillary Information System to Meet Future Mission Needs*

[2]: http://www.astronomy.swin.edu.au , last visit March 2009

[3]: R.R. Bate, D.D. Mueller, J.E. White, *Fundamentals of Astrodynamics*, Dover Publications

[4]: Bureau International des Poids et Mesures (http://www.bipm.org), last visit February 2009

[5]: http://www.eumetsat.int , last visit March 2009

[6]: W.M. Folkner, J.G. Williams, D.H. Boggs, Jet Propulsion Laboratory, *The Planetary and Lunar Ephemeris DE 421*, Memorandum IOM 343R-08-003, 03/31/2008

[7]: F. Lyard, F. Lefevre, T. Letellier ,O. Francis, *Modelling the global ocean tides: modern insights from FES2004,* Ocean Dynamics Journal

[8]: http://www.oc.nps.navy.mil , last visit March 2009

[9]: PDS website (http://www.pds.jpl.nasa.gov), last visit March 2009

[10]: http://scienceworld.wolfram.com , last visit February 2009

[11]: Spice Tutorials (ftp://naif.jpl.nasa.gov/pub/naif/toolkit_docs/Tutorials), last visit March 2009

[12]: Tapley, J. Ries, S. Bettadpur, D. Chambers, M. Chengs, F. Condi, B. Gunter, Z. Kang, P. Nagel, R. Pastor, T. Pekker, S. Poole, F. Wang, *GGM02 – An improved Earth gravity field model from GRACE*, Journal of Geodesy (2005), DOI

[13]: Thales Alenia Space, Phase-0 Study of the Post-EPS Mission, October 2008 (internal report)

[14]: Wikipedia (http://www.wikipedia.en), last visit March 2009

[15]: J.G. Williams, D.H. Boggs, W.M. Folkner, Jet Propulsion Laboratory, *DE421 Lunar Orbit, Physical Librations, and Surface Coordinates*, INTEROFFICE MEMORANDUM, IOM 335-JW,DB,WF-20080314-001, 03/14/2008

➢ ACKNOWLEDGMENTS

This is perhaps the hardest chapter to write, but at the same time the easiest and the more expected: this means I've reached the end. Few steps separate me from the goal and I find myself thinking on the progress done. Reached the end, always we think about the beginning, how everything started: how a news, seen by children eyes, opened the doors of a world of curiosity and research. The news was that the Voyager probe had transmitted telemetric data from his remote position keeping moving away from the Solar System. A dossier dealing with Voyager missions followed the news. I paid particular attention to the golden record onboard the spacecraft, a trace of our existence in the universe. I was affected by at that time president Carter's speech: "We cast this message into the cosmos…Of the 200 billion stars in the Milky Way galaxy, some-perhaps many-may have inhabited planets and space faring civilizations. If one such civilization intercepts Voyager and can understand these recorded contents, here is our message: We are trying to survive our time so we may live into yours. We hope some day, having solved the problems we face, to join a community of Galactic Civilizations. This record represents our hope and our determination and our goodwill in a vast and awesome universe". I imagined how a non-human could see, read, hear, and interpret the message arrived from a far away point in the universe, discover the existence of other living forms, how to reach our planet studying that beams (later on I understand the meaning of that pulsar of known directions from our Sun). That was the sparkle that have brought me thus far. Keeping on growing my knowledge and my curiosity, I decided to convert my passion in my studies and future work.

I want to express my gratitude to my thesis supervisor, prof. *Marco D'Errico*. I remember how during the course of Astrodynamic encouraged us with his words: "You will not become common engineers…you will be engineers with an 'E' so big ", showing with his hands the dimension of a fictitious big 'E'. He continually moved us in doing our better, saying that we should have earned that big 'E'. With his enthusiasm, his inspiration, his criticism, and his great efforts to explain things clearly and simply, he has made me mentally grow and love orbital mechanics. I'm sorry that after this thesis I can't momentarily keep on my studies with him hoping to join him again in the future.

I'm grateful to the whole team met at CO.RI.S.T.A., in particular to my tutor, Dr *Maria Rosaria Santovito*, who, with her wide knowledge, understanding, encouraging and personal guidance, has been great value for me and has provided good basis for the present thesis. I'd like to thanks also Dr *Giulia Pica* for her company and our "tea break", Eng. *Gianni Alberti*, my "adoptive tutor", and Matlab Man, Eng. *Claudio Papa*, with his software assistance.

I'm deeply grateful to my special friends *Eliana* e *Sofia*, pillars of my university career and of my live. We have shared good times and epic exams, always with the same strength and will. They have supported me especially during last months, encouraging, bearing and helping me in overcoming several difficulties and moods. This thesis would not have been possible without them. During this training I've felt lost without their good cheer and company.

Thanks also to all my friends. A special thank goes to *Tatore's Angels* (Lorena, Lidia and Antonella) even though they are odious when they call me together with their screeching voices (this is not a vivid e real thank since they have paid me to be named in this acknowledgments!!!).

I thanks my *grandparents* for their support and their pride and interest for my studies. They always asks me what I'm studying and which exam I'm preparing: each time I try to let them understand with few and simple words. After all Albert Einstein said that you do not really understand something unless you can explain it to your grandmother!!!

In the end, just to give them the importance they deserve, I thanks my *parents* for what they do for me: my mother whose love is boundless and my father, with his ingeniousness and meticulousness, who is my role model. There are not words to thanks them enough…I will be always indebted with them.

Aversa, March 27 2009