445–TP–002–002

# Theoretical Basis of the SDP Toolkit Geolocation Package for the ECS Project

**Technical Paper**

**May 1995**

Prepared Under Contract NAS5–60000

**RESPONSIBLE ENGINEER**

| | |
|---|---|
| Peter D. Noerdlinger | 5/18/95 |
| Peter D. Noerdlinger, SDPS Toolkit Developer<br>EOSDIS Core System Project | Date |

**SUBMITTED BY**

| | |
|---|---|
| Larry Klein | 5/18/95 |
| Larry Klein, SDPS Toolkit Manager<br>EOSDIS Core System Project | Date |

Hughes Applied Information Systems
Landover, Maryland

# Abstract

The basis for the Science Data Processing (SDP) Toolkit algorithms for time transformations, spacecraft ephemeris and attitude access, coordinate system transformations (other than map projections), and celestial body access is presented, as well as for the tools that determine if an Earth point or celestial body is in the field of view (FOV). A few typical scenarios for the efficient use of suites of the tools are suggested. The impact of the designs and the implementation on accuracy is discussed. The need to maintain data files of up to date Earth orientation data and leap seconds are emphasized. Earth models are briefly discussed. The depth of the presentation for various topics is less when they are well described in the literature (e.g., precession, nutation) and greater when new algorithms are derived for this Toolkit (e.g., the Earth Centered Rotating (ECR) to geodetic transformation, sub satellite point velocity, and refraction). Areas still under development or requiring improvement are noted. Several cautionary remarks are given.

***Keywords:*** geolocation, pixel, field-of-view, universal, time, Julian, date, planetary, ephemeris, spacecraft, attitude, aberration

This page intentionally left blank.

# Contents

## Figures

## Tables

# 1. Introduction

## 1.1 Purpose and Scope

The purpose of this document is to describe the algorithmic basis of tools provided in the SDP Toolkit. In this document, we will be concerned with tools that provide the following functionality:

- spacecraft ephemeris and attitude data
- time transformations to and from spacecraft clock, Coordinated Universal Time (UTC), Toolkit Internal Time, and Barycentric Dynamical Time (TDB)
- celestial body locations
- Earth orientation information
- transformations among spacecraft, Earth Centered and orbital coordinates
- tools to locate pixels, define FOV's, and determine the look and sun angles

The calling sequences, description of input and output data sets, and usage examples are described in the *SDP Toolkit Users Guide for the ECS Project*, (11/94, 333–CD–001–002) and in The *SDP Toolkit Primer for The ECS Project* (URL– http://edhs1.gsfc.nasa.gov).

This document will discuss all the time, celestial body, spacecraft ephemeris, coordinate transformation, geolocation and field of view tools, providing a basis for each. When new algorithms have been developed and are embodied in the tool, or in subsidiary functions used by the tool, the development herein is more extensive. In a few cases the reader is referred to the literature for tables of data that are embedded in or provided with the software, and which will not be duplicated here. In some cases, there may be reference to a function that is not given a full entry in the User Guide, i.e., the functions described in the Users Guide are the user interface only application program interface (API). The Toolkit is hierarchical, containing many lower level functions not directly accessible to the user.

It is expected that users of the tools will include developers of software for processing Earth Observing System (EOS) instrument data from raw counts to higher levels. For example, many users will want to geolocate pixels for subsequent calibration. Other users will want to re–geolocate data when better algorithms or calibration data are available in later mission phases. A variety of time transformations are provided for converting platform and instrument time stamps into other time systems in common use in the geophysics community.

## 1.2. Derivation of the Algorithms and Software

The Toolkit Geolocation, Celestial Body and Time Conversion Software is comprised of program units written at Earth Observing System Data and Information System (EOSDIS) Core

System (ECS); units obtained and/or translated from other sources, and data files from the United States Naval Observatory (USNO) and the International Earth Rotation Service (IERS). The present document provides derivations of all algorithms derived by Earth Observing System Data and Information System (EOSDIS) Core System (ECS) for implementation in the Toolkit. Derivations or brief sketches of heritage algorithms that have been imported are presented with literature references.

## 1.3 Suggested Usage of the Package: Time, Celestial Body, and Geolocation Tools

The Toolkit provides a large number of tools for usage in meeting requirements for production of scientific data products from Earth Observing System (EOS) instrument data. In this section, we give an overview of the usage of the package. For definitions of specialized terms, see Section 2.1. The symbols used in this document do not necessarily correspond with names used in the software.

### 1.3.1 Setup

The user is expected to set up a processing run as follows:

- Ensure that all necessary software and data files are present. See the User Guide under specific tools for these requirements. Typically, the user will need spacecraft ephemeris data and Earth model data to be present, and, in most cases, up to date tables of leap seconds and UT1/polar motion. If the sun or moon angles are desired, or the user needs the location of a planet, the Celestial Body ephemeris will be needed.
- Have all pixel or instrument field of view (FOV) location data available in spacecraft (SC) coordinates. Thus, the user will deal with gimbal angles, biases, and other items that are defined in the SC reference frame.
- Put all times into either individual UTC ASCII times, or bundles of times consisting of a UTC starting time in American Standard Code for Information Interchange (ASCII) and offsets in Standard International (SI) seconds, as double precision numbers. When the “bundled” method is used, most other input data must be supplied as arrays matched to the times, as specified in the User Guide. (Certain input data, such as the spacecraft ID tag, are assumed to be constant within a processing run.) Consult the User Guide for specifics.

### 1.3.2 Typical Geolocation Processing

Broadly, most of the geolocation issues will deal with finding the latitude and longitude of a pixel center, finding the "footprint" of a field of view (FOV), or finding when a selected Earth point or celestial body is in the FOV. The observer will also want some other information, such as the sun angle, and for night time observations, possibly the moon angle. Almost certainly, the zenith angle and azimuth of the observing ray itself will be of interest. The accompanying flow chart, Figure 1–1 suggest useful ways to intercombine the tools; but preliminary to setting up a run, it may prove desirable to set up arrays of time values (also see Section 4.7.1).

The geolocation tools generally accept times as a base time in UTC ASCII format plus offsets in SI seconds. Users having the times in some other form should convert them to this format. If it is desired to use a single UTC time the offsets can be omitted. Users wishing to process for a number of data packets should first convert packet times to a UTC base time as an array of offsets with the tool PGS_TD_SCtime_to_UTC(). The packet times would come from Level 0 data.

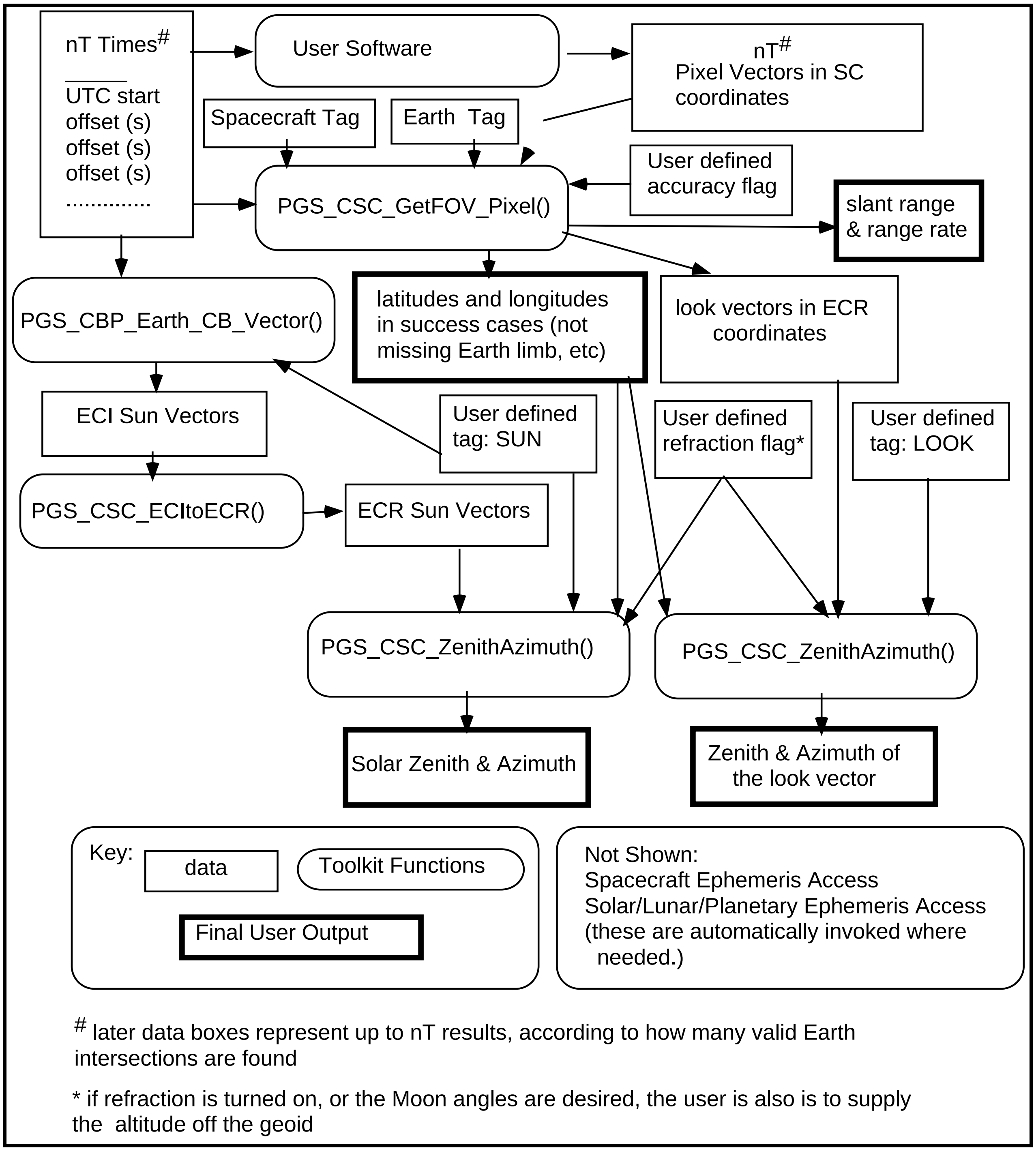

nT Times#
UTC start
offset (s)
offset (s)
offset (s)
..............
User Software
nT#
Pixel Vectors in SC coordinates
Spacecraft Tag
Earth Tag
User defined accuracy flag
PGS_CSC_GetFOV_Pixel()
slant range & range rate
PGS_CBP_Earth_CB_Vector()
latitudes and longitudes in success cases (not missing Earth limb, etc)
look vectors in ECR coordinates
ECI Sun Vectors
User defined tag: SUN
User defined refraction flag*
User defined tag: LOOK
PGS_CSC_ECItoECR()
ECR Sun Vectors
PGS_CSC_ZenithAzimuth()
PGS_CSC_ZenithAzimuth()
Solar Zenith & Azimuth
Zenith & Azimuth of the look vector
Key:
data
Toolkit Functions
Final User Output
Not Shown:
Spacecraft Ephemeris Access
Solar/Lunar/Planetary Ephemeris Access
(these are automatically invoked where needed.)
# later data boxes represent up to nT results, according to how many valid Earth intersections are found
* if refraction is turned on, or the Moon angles are desired, the user is also is to supply the altitude off the geoid

***Figure 1–1. Tool Flow Chart***

The problem of geolocating an individual pixel is straightforward; our algorithms provide directly the latitude and longitude of the look point and the zenith angle and azimuth of the sun and the look vector there.

The method for handling the footprint problem has been simplified in view of the many different instruments, with different shaped fields of view and different response characteristics. Some instruments are staring (field of view fixed in the spacecraft coordinates), some slewing, most are always pointed at the Earth, but some pass off the Earth limb routinely, or for calibration. To map all the resulting fields of view on terra firma is a formidable task. In addition, the appearance of the FOV on the Earth will in general be an irregular shape, quite sensitive to spacecraft and instrument attitude. Part may fall off the Earth, part on. Therefore we have handled FOV problems in two ways:

a. Individual look vectors in SC coordinates are mapped accurately onto the Earth by PGS_CSC_GetFOV_Pixel(). An array of these vectors (which can all be processed in one call) can be used to define the FOV perimeter; or a grid can be used to map the whole FOV on the Earth, in terms of latitude and longitude.

b. All other FOV problems are handled in SC coordinates. There, the FOV has a regular shape, as specified by the user through a family of ordered “perimeter vectors” that define the perimeter. The tools PGS_CSC_Earthpt_FOV() and PGS_CBP_body_inFOV() will determine if an Earth point of known latitude and longitude or a given celestial body is in the FOV at a specified time.

Next these two sets of problems—individual pixels and fields of view, are considered in turn.

### 1.3.3 Individual Pixels

This case includes arrays of pixel vectors. The suggested usage is as follows:

- The user supplies a spacecraft tag and Earth model tag (see Section 6.2.5.1), a starting time in UTC, an array of time offsets in seconds, and for each time a look vector in SC coordinates. Normally this will be a unit vector, but any nonzero vector in the right direction can be used. The user also ensures that the leap seconds and UT1/polar motion files are in place and up to date. Low or high accuracy is specified. In the high accuracy case, one should specify the coordinates of the center of the instrument aperture in SC coordinates (all zeros are acceptable).

#### 1.3.3.1 Latitude and Longitude; Slant Range and Range Rate

Tools:

- PGS_CSC_GetFOV_Pixel()

The tool PGS_CSC_GetFOV_Pixel() returns for each time the latitude and longitude of the lookpoint, the slant range and range rate, and, importantly the ECR look vector. In case of failure (look vector missing Earth) the latitude and longitude are set to $1.0 \times 10^{50}$ (see Section 2.3.1) and the warning PGSCSC_W_MISS_EARTH is set. This message will appear in the message log but may be overwritten as a return value if certain other problems exist.

### 1.3.3.2 Look Vector Aspect: Zenith and Azimuth

Tools:

- PGS_CSC_ZenithAzimuth()

To get the zenith and azimuth of the look vector, the user simply takes the Earth Centered Rotating (ECR) look vector (pixel vector) from PGS_CSC_GetFOV_Pixel() and passes it, with the latitude and longitude (from the same source), to PGS_CSC_ZenithAzimuth(). (The vectorTag identifier, a required input, is set to: PGSd_LOOK.) If the azimuth is not needed the flag for azimuth may be set to PGS_FALSE for efficiency. The user must also do the following:

a. Specify whether or not refraction is to be included. If refraction is included, the altitude must be entered consistently with a mean troposphere, because it will be used to estimate the sea level air density for the index of refraction.

b. Either set the altitude = 0, or, if refraction is to be calculated, or the moon angle is desired, use an appropriate model to get the altitude. For example users working with oceans might use a geoid model, those working with land data a digital elevation model (DEM), and instrument teams looking at the top of the troposphere, a troposphere model. Of course, anyone might encounter clouds, in which case the cloud altitude could be used. Fortunately, the zenith and azimuth are only slightly sensitive to altitude.

c. Check for $1.0 \times 10^{50}$ values indicating that look vector misses the Earth; bad ephemeris data; or other errors. Check for warning messages.

The zenith and azimuth will be returned as radians; the azimuth is measured East from North. If refraction is requested, the zenith angle will be that of the refracted ray—nearer to zenith than the unrefracted ray. The magnitude of the decrease in zenith angle is also returned for user convenience in checking.

### 1.3.3.3 Zenith and Azimuth Of The Sun Or Moon; Specular Reflections

Tools:

- PGS_CBP_Earth_CB_Vector()
- PGS_CSC_ECItoECR()
- PGS_CSC_ZenithAzimuth()

Now suppose that it is desired to know the zenith and azimuth of the sun vector, and the angle between the sun and look vectors. The tool PGS_CBP_Earth_CB_Vector() is called for the sun (the vectorTag identifier is set to: PGSd_SUN). The resulting Earth Centered Inertial (ECI) sun Vector is processed through PGS_CSC_ECItoECR(), to get the ECR sun vector. The ECR sun vector is then passed to PGS_CSC_ZenithAzimuth() along with the latitude, longitude and, if required, the altitude, to get the solar Zenith and Azimuth. As before (with the Look Vector) refraction may be turned on or off; but if on, then the altitude must be specified; while if off, the altitude is ignored. The user can obtain the angle between the sun vector and look vector by transforming the sun vector from ECI to ECR (which may have been done already if the sun zenith angle was determined), normalizing it, and taking the inverse cosine of the negative of their dot product (the negative sign is needed because the two vectors have opposite senses of definition—the look vector toward Earth, the sun vector away). The refraction correction cannot be implemented in this procedure; it would require user written software to refract the sun vector toward the zenith by the change in angle. (It is relatively easy to do this by constructing the surface normal and taking a linear combination of the two vectors; the surface normal is easily obtained from the latitude and longitude.) Alternatively, the two zeniths and azimuths after refraction can be used with formulas from spherical geometry to find the angle. If only the angle between their vertical planes is desired, the user may simply difference the azimuths of the sun and the look vectors.

Exactly the same procedure can be followed to determine the zenith angle and azimuth of the moon. (The vectorTag identifier is set to: PGSd_MOON.) The function PGS_CSC_ZenithAzimuth() automatically accounts for the parallactic displacement of the moon vector due to topocentric geometry.

If it is desired to look for specular reflection, as off a pond, a horizontal ice sheet, or the sea, the user simply requires that the zenith angles of the look vector and the sun (or moon) vector be equal (within some tolerance based on surface slope and roughness) and that their azimuths be $\pi$ radians different (again within some tolerance). The zenith and azimuth angles are found from PGS_CSC_ZenithAzimuth(), using sun and moon vectors from the tool PGS_CBP_Earth_CB_Vector(), and transforming to ECR.

### 1.3.4 FOV Based Tools

Tools:

- PGS_CSC_Earthpt_FOV()
- PGS_CBP_body_inFOV()

These tools determine if an Earth point or a celestial body is in the FOV. The point about which it is asked, "is this in the field of view," is called the "candidate point." Both tools require the FOV to be specified as an ordered sequence of vectors pointing at its periphery, in SC coordinates. A "center" vector in the FOV and preferably near the center must also be supplied. It is used to define the center versus the outside; geometrically, but, more importantly it is used as the center of a "conical hull" that the tools circumscribe around the FOV.

These tools can be called independently of the others, so no flowchart is provided. Their overall operation is sketched here and details are given in Section 3 below.

In all cases, the candidate point is tested first to see if it is in the conical hull, because this test is fast. In the case of CB (celestial body) in FOV, a test for Earth blockage of the conical hull is also applied first to avoid executing a complicated algorithm when the FOV cannot possibly be looking at a CB.

The Earth point in FOV tool will ascertain if an Earth point of known latitude, longitude and altitude is in the FOV. If so, it also returns, as a convenience, the unit vector towards the point in SC coordinates. This can be used to locate the candidate point within the FOV by user software written in SC coordinates.

The CB in FOV tool accepts input tags that can be set for the usual toolkit Celestial Bodies—the sun, moon and the planets other than Earth. It accesses the planetary ephemeris, and returns a flag to tell the user if the CB is in the FOV. It also returns the vector to the CB in SC coordinates, to facilitate user checking or further processing. (For example, users wishing to know exactly where the CB is within the FOV will not have to call the Celestial Body ephemeris anew—they can compare the CB vector in SC coordinates with a map of the FOV in the same system.) The tool also allows for input of an arbitrary user–defined "CB"—which could be a bright star—in that case the tag should be set to "PGSd_STAR" and the ECI coordinates of the object must be supplied by the user. The vector to the CB in SC coordinates is still returned.

The radii of the celestial bodies are set up to include the bright satellites in the case of planets—see the table under the detailed discussion of that tool (in the case of Pluto, Charon is not very bright but is comparable to the planet).

## 1.4 Accuracy and Validation Issues

The accuracy of the individual functions described above has been tested by comparison to tables in the Astronomical Almanac and against EOSAT/Moderate–Resolution Imaging Spectroradiometer (MODIS) heritage software. The remainder of this section discusses the accuracy based on errors in the data as well as algorithmic roundoff/truncation error.

This document necessarily refers to the propagation of certain kinds of error through the algorithms and software. In order for these references to be meaningful, the discussion will begin with an analysis of sources of error. Each source of error will undergo a translation from the original measure into meters at the Earth's surface. For example, an error of so many seconds arc in Earth rotation may be translated directly into meters, but an error in the Earth's axial rotation may originally be expressed in seconds of time, radians, or fractions of a Julian day. Such an error is first translated to angular measure using the Earth's angular rotational velocity. Again, a time error in the spacecraft ephemeris translated into meters of spacecraft motion using its velocity. In this document, to analyze the latter error, we make a simple linear flat–Earth approximation and assume that; 50 meters error in the spacecraft position results in 50 meters error at the Earth's surface. In analyzing errors related to angular measure about the Earth's center, we generally assume the equatorial radius, although an error in rotation about the Earth's axis results in a smaller linear error at high latitudes.

## 1.5 Error Sources

This section discusses all known geolocation error sources. It emphasizes geometric issues; for time related issues see Section 4.7.2. To put different error sources on a common scale, all are converted to equivalent meters geolocation error. To convert certain errors, such as those in time or angle, to equivalent meters, it is necessary first to present some conversion factors.

### 1.5.1 Constants and Conversion Factors

The following table presents some "constants" for the reader's convenience in estimating scale changes under some of our transformations. Some of the numbers are only representative of slowly changing values that may be considered nearly constant over periods ranging from a day to the mission lifetime. In all cases, the most accurate available value of the "constant" has been used. For example, the Temps Atomique International (TAI) day will be kept at 86,400 SI seconds for the foreseeable future, but the sidereal day changes very slowly due to the motion of the equinoxes, nutation, and lunar torques. In principle, the conversion of UT1 to Greenwich Mean Sidereal Time (GMST) (q.v. below) is mostly a matter of converting Earth rotation from a solar to a sidereal system, but the actual conversion is not based on a mean value such as shown in Table 1–1. Instead Product Generation System (PGS) software uses expressions from the International Astronomical Union (IAU), the U.S. Naval Observatory, or the International Earth Rotation Service (IERS). as appropriate for any important conversions; references are given in the individual sections.

***Table 1–1. Conversion Factors***

| NAME | VALUE |
|---|---|
| arc seconds per radian | 206264.8 |
| radians per arc second | 4.848E–6 |
| degrees/hour (Earth rotation) | 15 |
| degrees/hour (hour angle) | 15 |
| arc seconds/second of time (Earth rotation or hour angle) | 15 |
| (meters at Earth's surface)/(arc second Earth rotation) | 30.1 |
| Earth rotation rate (sidereal) | 0.000072921151467 rad/sec |
| Earth equatorial surface velocity in ECI | 465.1 m/s |
| mean sidereal seconds per solar second | 1.00273790935 |
| d(UT1)/d(TAI) in late 1994 (approx) | 0.999999974 |

### 1.5.2 Sources of Error

Table 1–2 shows typical errors in the Earth motion and time data, and their conversion to meters of geolocation error. We first discuss other sources of error—generally ones about which the Toolkit can do nothing, or ones that are already compensated in it.

### 1.5.3 Ephemeris/Attitude Errors

Error in the spacecraft ephemeris (perhaps 100 to 200 m) and attitude (perhaps several arc seconds) are likely to dominate initially. Our purpose is to keep the errors introduced by errors in Earth motion, nutation, roundoff and algorithmic approximations negligible as compared with such sources; and at the level of < 1 meter equivalent position, absolute, and a few cm in smoothness. In this way, users wishing to use control or tie points, or related methods to improve the orbit or attitude data, or compensate biases, will not be hindered. The effects of errors in ephemeris and attitude have been thoroughly analyzed, e.g., by the Multi–Angle Imaging Spectro–Radiometer (MISR) team and will not be discussed here.

### 1.5.4 Roundoff Error

The truncation or roundoff error per double precision floating point operation was determined by intercomparing answers on different platforms and by using the USNO test program “CDEEP.c”. That program is available by ftp (file transfer protocol) and by e–mail from the Observatory. One can use anonymous ftp from tycho.usno.navy.mil (user = anonymous, password = guest), or can send e–mail to adsmail@tycho.usno.navy.mil. The standard platforms for the Toolkit are all 32 bit machines, which means 64 bits are used for double precision. By using CDEEP, it was determined that all the standard workstations have an equivalent decimal precision of better than 1.2 parts in $10^{16}$ per double precision division operation. The Cray C90 computer that was used for some early testing had an error of 3.6 parts in $10^{15}$ when ordinary C double precision variables were used, but Cray has a “long double” type available, which is 128 bits, that yields truncation error of less than a part in $10^{29}$. It is planned to set the Toolkit typdef “PGSt_double” to “long double” on the Cray in the future.

Typically, a Toolkit tool will have 5 to 30 operations chained in sequence to obtain a single vector component, latitude, or longitude. Interplatform comparisons on several functions gave agreement within 1 few parts in $10^{15}$, as expected. By using double precision variables, and in a few cases, two doubles in tandem (for Julian Dates), the truncation or roundoff error in the Toolkit has been kept smaller than all other errors. (See detailed discussion in the section on time.)

### 1.5.5 Interpolation Error

Additional error will be introduced in interpolating the ephemeris and attitude. The Toolkit implementation at the time of this writing has only preliminary tools for doing this; i.e., the attitude interpolation is linear. Orbit simulation software from Computer Software Management and Information Center (COSMIC), which will facilitate testing the cubic ephemeris interpolation is being used for testing. Other interpolation methods are currently being considered. Interpolation error for the Earth motion routines has been kept smaller than any other error. See Section 4.9.1.2.

### 1.5.6 Earth Orientation Parameters (EOP) Error

The 1 meter absolute error is unavoidable unless EOP data are included, but since Goddard Space Flight Center (GSFC) Flight Dynamics Facility (FDF) does not use EOP data (a measure of the error in the 1980 IAU nutation theory) there is no point to using these data in the Toolkit. Doing so would introduce a systematic offset in the Spacecraft ephemeris of up to 1 meter, varying on a time scale of several months to a year.

### 1.5.7 Miscellaneous Errors

Errors due to errors in the figure of Earth, ocean tides, Earth tides and continental drift are not discussed here, although the section on Earth models lists some references for tidal data and station coordinates.

The correction for aberration in the pixel lookpoint locator and in the ECI to SC and SC to ECI transformations is essential at the accuracy level of 5 to 35 meters for users wishing to use control points, if the same point is likely to be viewed both on North and South passes, or at quite different angles to the velocity—see the sections on aberration. Because the subsatellite point function does not include aberration, that point will be geometric only and will disagree with the lookpoint of a nadir pointing instrument by about 8 m for the Tropical Rainfall Measuring Mission (TRMM) to 16 m for the AM or PM spacecraft. This is not an error, but a discrepancy.

### 1.5.8 Refraction Errors

Typical errors in geolocation of about 18 m at 60 deg zenith angle and 70 m at 70 degrees zenith angle can be reduced several fold. Refraction compensation for the zenith angle of the sun and the look vector is available in the current implementation, but only in white light. See Table 6–5.

***Table 1–2. Typical Error Values in Underlying Data***

| NAME | TYPICAL Value | TYPICAL 1 sigma ERROR |
|---|---|---|
| UT1–UTC | up to +- 0.9 sec | 0.0001 s (up to 0.0003 in 1988, etc.) |
| x ,y (Polar motion) | +- 0.4 arc sec | 0.0008 arc sec |
| Nutation angles | +- 9 arc sec | 0.025 arc sec (EOP data) + 0.0001 random |
| Earth rotation during light travel | max 3.7 m - AM1 | can be compensated |
| Earth rotation during light travel | max 1.1 m - TRMM | can be compensated |

Each correction that is in arc seconds can be translated to meters equivalent for geolocation by multiplying by the factor 30.1 from Table 1–1. Errors in time are translated with Earth rotation. Table 1–3, shows the equivalence in meters. The uncompensated case corresponds to missing, out of date, or corrupt data files (Earth motion). Ignoring the EOP part, the uncompensated case for nutation could occur only if there was an internal Toolkit error or a wrong date was supplied, because the nutation routines are invoked automatically where required.

***Table 1–3. Translation of Errors to Equivalent Meters***

| NAME | Error if uncompensated | Typical 1 sigma error after compensation |
|---|---|---|
| UT1–UTC | up to +- 420 m | 0.05 m (up to 0.15 in 1988, etc.) |
| x ,y (Polar motion) | +- 12 m | 0.024 m |
| Nutation angles | +- 270 m | 0.75 m (EOP data) + 0.003m random |

One sees that although the UT1 data are good, they must be kept up to date—see the section on time.

The errors in UT1 and polar motion are not identified in the original files as 1 sigma or 3 sigma, but Dr. Dennis McCarthy, Head of the Earth Orientation Department at the United States Naval Observatory (USNO), has informed us that they are 1 sigma.

## 1.6 Organization

The discussion in this introduction is amplified and extended in the remainder of the document. The organization is as follows:

***Table 1–4. Document Organization***

| Section | Description |
|---|---|
| 1. Introduction | |
| 2. Fundamental Assumptions | assumptions made in tool development |
| 3. Spacecraft Ephemeris and Attitude | description of Toolkit handling of SC ephemeris inputs |
| 4. Time Streams and Time Transformations | time formats and comparisons between them |
| 5. Celestial Body Access | discussion of coordinate systems, parallax, aberration and other consideration in celestial body position access |
| 6. Coordinate System Conversion | discussion of coordinate systems, e.g., spacecraft to Earth–centered inertial, etc., and transformations used by the Toolkit |
| 7. Pixel and Sub–Satellite Point Tools | discussion of Toolkit acquisition of sub–satellite point, determination of pixel location, etc. |
| 8. Earth Point or Celestial Body in FOV Tools | discussion of instrument field of view issue |
| 9. Economies and Shortcuts | shortcuts and approximations used by the Toolkit |
| Appendix A | discussion of leap seconds file |
| Appendix B | Julian day conversions |
| Appendix C | polar motion data and corrections |
| Appendix D | calculations and constants used by the atmospheric model |
| Appendix E | Celestial body models and discussion |
| Appendix F | description of test of a point inside an instrument FOV |
| Appendix G | description of the Jet Propulsion Laboratory (JPL) planetary ephemeris file |
| Appendix H | calculation of the curvature in the Earth in the plane of the meridian |
| Appendix I | The Barycentric time correction for Earthbound clocks |
| Appendix J | listing and brief description of Tools in the Toolkit API |
| Abbreviations and Acronyms | |

Questions regarding technical information contained within this Paper should be addressed to the following ECS and/or GSFC contacts:

- ECS Contacts
    - Peter Noerdlinger (pnoerdli@eos.hitc.com)
    - Ed Larson (elarson@eos.hitc.com)
    - Larry Klein (larry@eos.hitc.com)
- GSFC Contacts
    - Dan Marinelli (dan@marinelli.gsfc.nasa.gov)

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Applied Information Systems
1616 McCormick Dr.
Landover, MD 20785

This page intentionally left blank.

# 2. Definitions and Fundamental Assumptions

## 2.1 Definitions

Certain definitions, such as those of the coordinate systems, or Julian Dates, require separate sections, but in the present section a brief list of terms is presented such as can be explained in a few words.

- aberration—the difference in the direction of a light ray as measured in two reference frames in relative motion
- altitude—height in meters off the ellipsoid (except in PGS_CSC_ZenithAzimuth(), where it is off the geoid.)
- boundary arc—a portion great circle on the sky connecting two perimeter vectors (q.v.)
- Doppler velocity—speed of the look point terrain relative to the instrument, projected on the line of sight, positive if “away”, negative if “toward.”
- Earth model tag—see “ellipsoid”
- ECI coordinates—the rectangular, Earth centered, inertial coordinate system J2000 (Section 5.2)
- ECI look vector or pixel vector—the look vector in ECI coordinates
- ECR coordinates—a rectangular, Earth fixed coordinate system (Section 6.2.2)
- ECR look vector or pixel vector—the look vector in ECR coordinates
- ellipsoid—the spheroid selected by the user as a model of the Earth, and identified by an Earth model tag (Section 6.2.5.1)
- field of view (FOV)—the portion of the sky within the instrument aperture (see “perimeter vectors”)
- geoid—an equipotential surface of gravity whose average position is the ellipsoid
- inFOV vector—a user supplied unit vector within the field of view—required for tools that test points to see if they are in the FOV.
- J2—a dimensionless measure of the gravitational effect of the Earth's equatorial bulge
- Julian century—36525 days[1]

[1]Resolution C7 of the IAU (1994) defines the Julian Century as 36525 days of Terrestrial Time (essentially equivalent to TAI). Since TDT differs by only a constant, and TDB by additional small periodic terms, it is, for

- “latency” and “latent data” refer to out of date data or data files
- latitude—unless otherwise qualified, it is always geodetic latitude in radians (Section 6.2.5)
- leap second—a jump (normally backwards) taken in the UTC clock to keep it near UT1
- leap second interval—the time interval of a positive leap second, during which the UTC clock runs past 24 hours, without incrementing the day, normally just before midnight
- longitude—measured East from Greenwich in radians (negative values, or from $\pi$ to $2\pi$ are West)[2]
- look vector—a vector along a line of sight from the instrument, such as the boresight or any point within the field of view. Can be a unit vector or have components in meters.
- look point—he intersection of the look vector with the Earth ellipsoid. (In some functions an altitude off the ellipsoid or geoid is allowed for on input.)
- parallax—the apparent displacement of an object due to the displacement of the observer
- perimeter vectors—an ordered set of vectors that define the FOV in SC coordinates. None may be 90 degrees or more from the inFOV vector (q.v.)
- pixel vector—same as look vector
- quaternion—a set of 4 real numbers, subject to a certain algebra, that defines a rotation. See Section 3.6.1
- return value—the integer value returned by a Toolkit function, or it’s equivalent as a message (see the *User Guide* section on the SMF tools)
- slant range—distance in meters from instrument to look point
- sub satellite point—the point on the ellipsoid at the base of a normal to it from the satellite
- TOD coordinate system—the rectangular, Earth centered, inertial coordinate system with its pole along the rotation pole of date (Section 5.3)
- topocentric—a geometry and coordinate system set up at a terrestrial point, with the Z axis at zenith, the X axis North, and the Y axis East

---

practical purposes, permissible to use the round number 36525 days also for TDT and TDB. This will be consistent with previous practice for ET.

[2]On input, the Tools accept any value for longitude, but it is recommended to use the range $(-\pi,\pi)$, which is used on output. Obviously, longitudes that differ by an integral multiple of $2\pi$ refer to the same location. In the author’s experience, some function libraries deteriorate in accuracy when supplied very large arguments for trigonometric functions.

- unit vector—a vector whose length is 1.0. To avoid problems with platforms of different accuracy the Toolkit functions generally accept any nonzero vector, whose length is ignored
- zenith—the direction normal to the Earth ellipsoid, generally at the look point or subsatellite point
- zenith angle—the angle of a vector from the zenith, in radians

## 2.2 Assumption Basis

The algorithms implemented in this Toolkit are based on several assumptions:

- SI units are used throughout, and all angles are in radians.
- For AM and PM the spacecraft Ephemeris is in J2000.
- The TRMM spacecraft Ephemeris is in ECI True of Date coordinates[3].
- The user supplies look vectors in Space Craft (SC) coordinates, with times in UTC + offsets in seconds.
- Times not earlier than June 30, 1979 are to be used, unless user replaces the UT1–UTC and polar motion table, *"utcpole.dat"*, with one obtained by reformatting the delivered file "*utcpole.1972to1979*" ) (see Appendix C)
- The user is responsible for bias, scan mode, mechanical, optical, glint, and related problems.
- Files of UT1–UTC, Polar Motion and leap seconds are acquired and maintained by ECS.
- The user will trap condition output value = $1.0 \times 10^{50}$, and monitor Toolkit error and status message logs. (See Section 2.3.1)
- The Earth model used in Toolkit algorithms is spheroidal (i.e., an ellipsoid with two of the three major axes equal).
- Aberration and related effects are calculated only to order v/c, where v is the relative velocity and c that of light..
- No effects of general relativity are included except in the difference Barycentric Dynamical Time (TDB)–Terrestrial Dynamical Time (TDT).
- The DE200 ephemeris is valid.

## 2.3 Default Decisions and Outputs on Nonstandard Inputs

In designing and writing the software, every effort has been made to anticipate cases where inputs may not jibe with expectations, or where data files may be missing, so as to impair the

[3]In TK4 it was assumed on the basis of certain TRMM documen ts that the ephemeris would be ion J2000

processing. The philosophy has been to attempt to continue processing but to issue warning messages and/or returns. This section discusses the options taken in several typical cases that are likely to be encountered in practice, for the Ephemeris Data Access (EPH), Coordinate System Conversion (CSC), Time and Date (TD), and Celestial Body Position (CBP) tools. Generally, individuals coding the various tools and functions have made, from time to time, decisions that may not have been reviewed by the lead staff, so that in the more unlikely conditions, such things as the precedence of different error messages may vary from tool to tool.

“Nonstandard Inputs” in this section are meant to be any of the following:

- improperly normalized vectors supplied in the calling list when a unit vector is required
- out of range input values
- missing files for spacecraft ephemeris, figure of Earth, leap seconds, or UT1/polar motion, or out–of–date files in the last two cases)
- improper tags for spacecraft identification, Earth model, celestial body, etc.

“Outputs” in this section are meant to be any of the following:

- answers returned in the calling list
- return values
- messages written to the log file in $PGSRUN/LogStatus (see the User Guide for configuration details)

The term “Nonstandard Input” will hereinafter be abbreviated as “bad input” without pejorative intent.

### 2.3.1 Overall Methodology

In several cases, such as an invalid spacecraft tag, missing spacecraft ephemeris, or time format error, no processing can take place, and processing is ended with an error return. Many of the geolocation tools work on arrays of inputs, with an array of time offsets defining the times. In these cases, every attempt is made to continue processing data points even after a bad point is encountered. Double precision output values for points that cannot be processed are set to $1.0 \times 10^{50}$, so that the user can recognize these. Integer and Boolean values may not be meaningfully populated on output; although in PGS_CSC_DayNight() the Boolean return is assigned the error status code in case of failure. When a tool works on single input point, even the double precision output values may not be populated in case of a bad input.

Since each function has only one return value, messages are generally written to the log file in case of problems. In most cases, the identity of the “offending” (bad input value) point is encoded into the message, for example by giving its sequence number in the input array. In case of a serious problem on one point, an attempt has been made to save the serious warning or error as the return value if a lesser problem occurs for a later point, but messages are generally written to the log file in any case. In a few cases, such as an attempt to find Earth intersection of a vector missing the Earth, the error is deemed to be so likely and frequent that the number of error messages written to the log file has been limited, to avoid excess growth of that file.

The error PGS_E_TOOLKIT is used when a Toolkit function encounters an impossible condition on return from a subordinate function. For example, if a time known to be in correct format is passed to a time tool, and that tool issues a return indicating a wrong format, then obviously the software is damaged, or data have been corrupted during processing. This return should be reported at once to the Toolkit point of contact.

In summary, there are three (3) ways that problems are reported: return status, messages to the log file, and values of $1.0 \times 10^{50}$ in the calling argument list of outputs.

### 2.3.2 Improperly Normalized Input Vectors

Depending on the tool, input vectors are sometimes dimensioned (in meters or meters/second) or they can be unit vectors. If dimensioned vectors are called for, there is no way to check the units. Also, ranges are generally not checked, although in some cases a warning is issued if the user appears to want to work with coordinates of a point deep within the Earth. In the case of a unit vector, the Toolkit had to face the problem that a user might supply a vector that had the intended direction but, for some reason, was not a true unit vector to machine accuracy. For example, the vector might have been normalized in single precision, or its components read from a data file in a format not preserving full machine accuracy. *Therefore, whenever a unit vector is called for, it is anyway copied to a scratch vector and normalized again before use.* At the same time, a check is performed so that if the vector is zero (0,0,0), then a warning issues and the rest of the calculation is omitted. In the cases PGS_CSC_ECItoSC and PGS_CSC_SCtoECI, the length of the input vector is instead tested against the range (0.99999, 1.00001) and certain decisions are based on the result—see the detailed descriptions for explanation.

### 2.3.3 Bad Earth Model Tag or Missing File “earthfigure.dat”

In these cases the WGS84 model is invoked and a warning message is issued, with a warning return PGSCSC_W_DEFAULT_EARTH_MODEL. The WGS84 values are encoded directly in the software for this purpose.

### 2.3.4 Miscellaneous

In the case of an invalid spacecraft or celestial body tag, no processing can be done for tools requiring such tags. Note that celestial body and spacecraft tags have PGSt_integer equivalents that may be looked up in the User Guide or the include files in $PGSINC. Earth model tags are strings, for consistency with the AA tools. See specific tools and Section 1.5 for the effects of missing or latent files of leap seconds, UT1, or polar motion.

This page intentionally left blank.

# 3. Spacecraft Ephemeris and Attitude

## 3.1 Introduction

This section describes the tools used to access and interpolate the spacecraft ephemeris and attitude.

Tools:

PGS_EPH_EphemAttit()

### 3.1.1 Organization

This section is organized as follows:

3.1—Introduction

3.2—Ephemeris Interpolation

3.3—Attitude Processing

3.4—Attitude Interpolation

3.5—The Toolkit TRMM and EOS AM1 simulators

3.6—Operations on quaternions and Euler angles

### 3.1.2 Summary

This tool group contains tools and associated software that provide access to the spacecraft ephemeris and attitude at a given time. Currently the EOS AM Project (morning spacecraft series) (EOSAM), EOS PM Project (afternoon spacecraft series) (EOSPM) and TRMM platforms are supported. In current implementation of the Toolkit, orbit and attitude data is supplied by the ECS Spacecraft Orbit and Attitude Simulator, which is based on Upper Atmosphere Research Satellite (UARS) Heritage code.

This simulator (orbsim) will create files of simulated spacecraft orbit and attitude data necessary to use the Toolkit spacecraft ephemeris and attitude data access tool (PGS_EPH_EphemAttit( )). Users may alternatively create their own data files but MUST follow the format described in the SDP Toolkit Users Guide for the ECS Project.

## 3.2 Ephemeris Interpolation

At present all ephemeris and attitude data come from the simulator. Error or noise in the position is not simulated. The present ephemeris interpolation method fits the nearest two position and velocity points component by component with a cubic polynomial in time. This method is excellent when the data are already smoothed and at frequent time intervals. When observational noise is present or the data are too widely separated, more sophisticated methods may be needed. Some of the usual methods, however, depend on assuming that the orbit is a Keplerian ellipse, which is not strictly correct for low orbits, because of perturbations. Alternate methods of interpolation are being assessed for possible future implementation in the toolkit.

In detail, the method used for orbit interpolation at this are as follows:

Each component of the orbit vector is considered as an independent function in one dimension, with the velocity as the first derivative. The position and velocity for each vector component are combined to perform cubic interpolation; this ensures that the interpolated positions and velocities are consistent. The interpolated positions are evaluated directly from the cubic and the velocities are computed using the derivative (a quadratic). If the time coordinates of the two end points are arbitrarily assigned values of 0 and 1, then the polynomial coefficients $a_0$, $a_1$, $a_2$, and a3 are computed as follows:

$$a_0 = P_1 \tag{3.2–1}$$

$$a_1 = V_1 \tag{3.2–2}$$

$$a_2 = 3*(P_2 - P_1) - dT*(2*V_1 + V_2) \tag{3.2–3}$$

and

$$a_3 = 2*(P_1 - P_2) + dT*(V_1 + V_2) \tag{3.2–4}$$

where $P_1$, $P_2$, $V_1$ and $V_2$ are the successive values of the position and velocity, respectively, and $dT = (T_2 - T_1)$ is the time difference in seconds. To interpolate to any intermediate point, the desired sample time $T_S$ is converted to a relative value between 0 and 1:

$$T = (T_S - T_1) / dT \tag{3.2–5}$$

The position and velocity are then computed from the cubic and its derivative, viz:

$$P = a_0 + a_1*T + a_2*T^2 + a_3*T^3 \tag{3.2–6}$$

and

$$V = (a_1 + 2*a_2*T + 3*a_3*T^2) / dT \tag{3.2–7}$$

I am indebted to Fred Patt of the General Sciences Corporation for the concept of this algorithm.

## 3.3 Attitude Processing

The toolkit accesses attitude data in a variety of ways. (See Level 0 Data Issues for the ECS Project, for details on TRMM, AM and PM attitude data access.) The input attitude is in all cases converted to quaternions that carry a vector from the SC coordinate system to ECI. We have designed the algorithms and software to handle TRMM, AM and the PM series, although the TRMM work is preliminary, pending further definition of TRMM platform specifications.

In most cases, spacecraft data are expected to come to the Toolkit in the form of Euler angles. The Toolkit, on the other hand, works with quaternions, because they are free of singularities. Section 3.6.1 shows how the Euler angles are converted to quaternions. These quaternions are used to transform between orbital and SC coordinates, because, for the spacecraft on which we have information thus far, the Euler angles are defined so as to relate these two systems. It would also be possible to use Euler angles to relate the spacecraft system to ECI, but for three–axis stabilized spacecraft that makes little sense, because the variations of angles then contain periodic terms of the orbital period.

Note that our overall attitude quaternions, following the convention of the attitude matrix in the UARS simulator, takes vectors from the spacecraft reference frame to ECI, while Wertz, for example defines it the other way in *Spacecraft Attitude Determination and Control*, Ed. J. R. Wertz (D. Reidel, Holland, 1978). The conversion of the Euler angles to quaternions is discussed in Section 3.6, after the attitude interpolation.

### 3.3.1 Attitude Processing for EOS AM and EOS PM

The attitude processing for AM1 and the PM series is shown in the Figure 3–1.

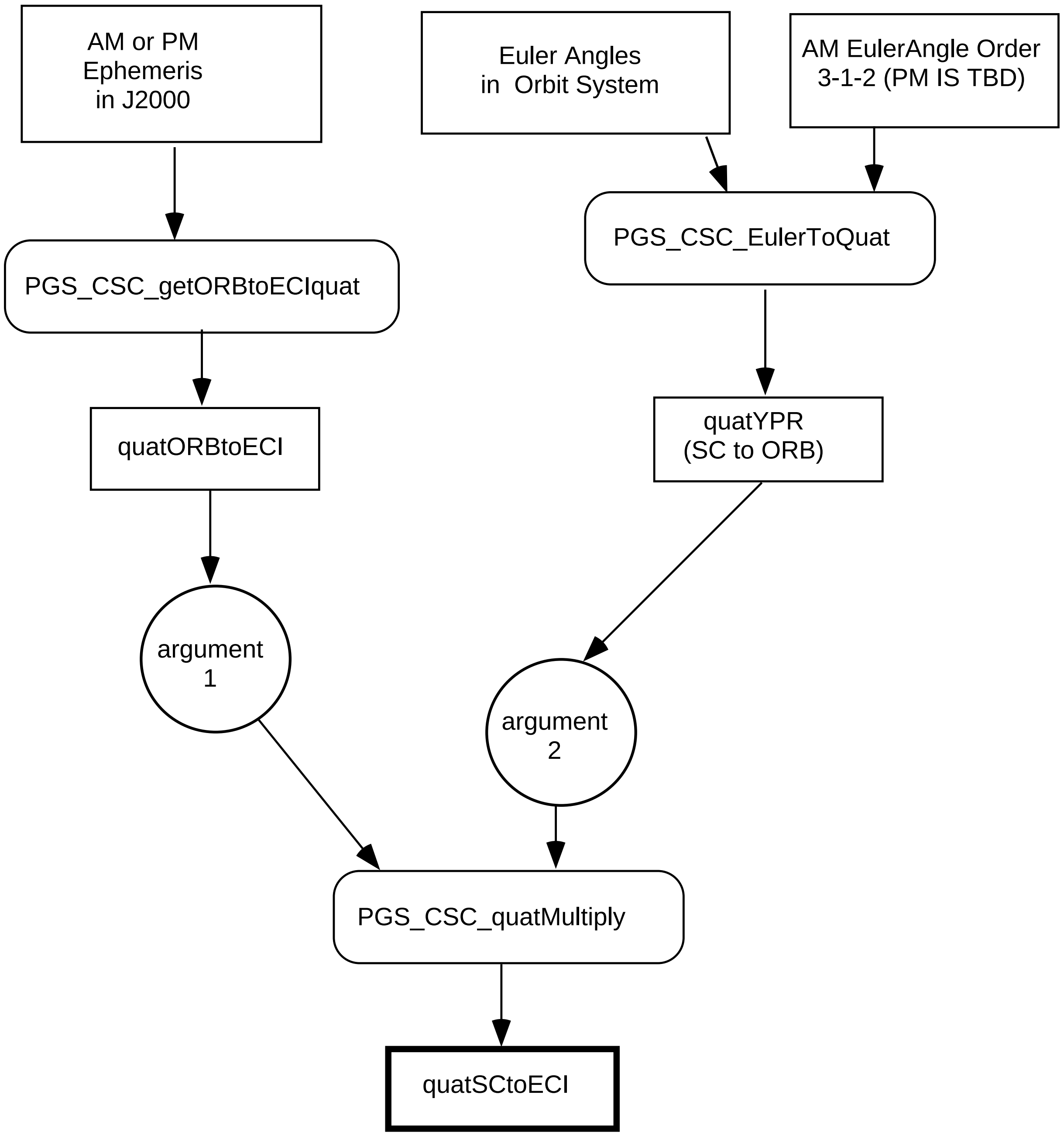


***Figure 3–1. EOS AM and EOS PM Attitude Processing***

Note that only the spacecraft position is needed to obtain the quaternions to transform between ECI and orbital coordinates, but the spacecraft attitude data are used for the transformation between spacecraft and orbital coordinates. The figure of the Earth makes no difference in this case, because these spacecraft are referenced to geocentric nadir.

### 3.3.2 Attitude Processing for TRMM

For TRMM, there is an additional coordinate system, the "Tipped Orbital" system, to which the spacecraft attitude is referenced. This system exists because the spacecraft uses Earth horizon sensors to find the roll and pitch. (A combination of sun sensor and gyro data are used to obtain the yaw.) The "Tipped Orbital" system is, in principle, an orbital system referenced almost to geodetic nadir. Figure 3–3 shows how North and South Earth limb vectors, estimated from horizon crossings, are used to construct a “tipped” coordinate system. The attitude processing for TRMM is shown in the Figure 3–2.

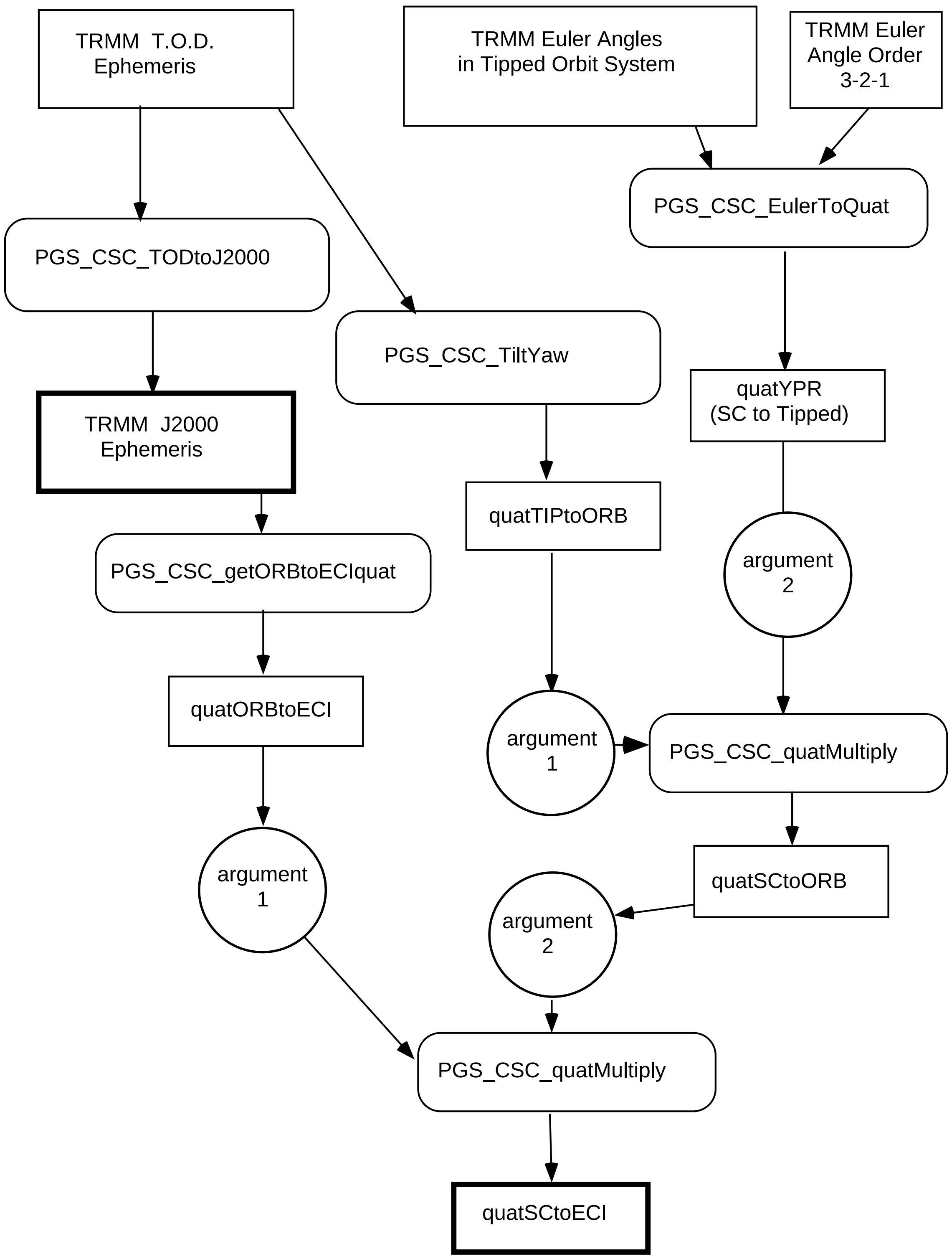


***Figure 3–2. TRMM Attitude Processing***

Figure 3–3 shows the relationship of the horizon tangent vectors; the bisector used to define “tipped” nadir; the true subsatellite point; and the vector between TRMM and Earth center. The TRMM attitude processing algorithms were developed and tested with close reference to this diagram, with tests performed to ensure that the vectors were, for the TRMM and even for hypothetical orbits, in the required order, as shown. Of course, the figure shows only a two dimensional representation of a three dimensional problem. The points S and P are within about an arc second of each other in latitude, which is consistent with FDF validation of the TRMM “tip angle” algorithm. The difference is biggest at intermediate latitudes. The angle of line OT to the equator is the TRMM geocentric latitude, and of ST the TRMM geodetic latitude. The angle of PA to the equator is just the geodetic latitude of P, and has no great significance except for nadir–pointing instruments, which will view P (ignoring aberration). The tools PGS_CSC_SubSatellitePoint() and PGS_CSC_GetFOV_Pixel() were used to find the points S and P respectively, with the aberration artificially turned off to get purely geometric tests.

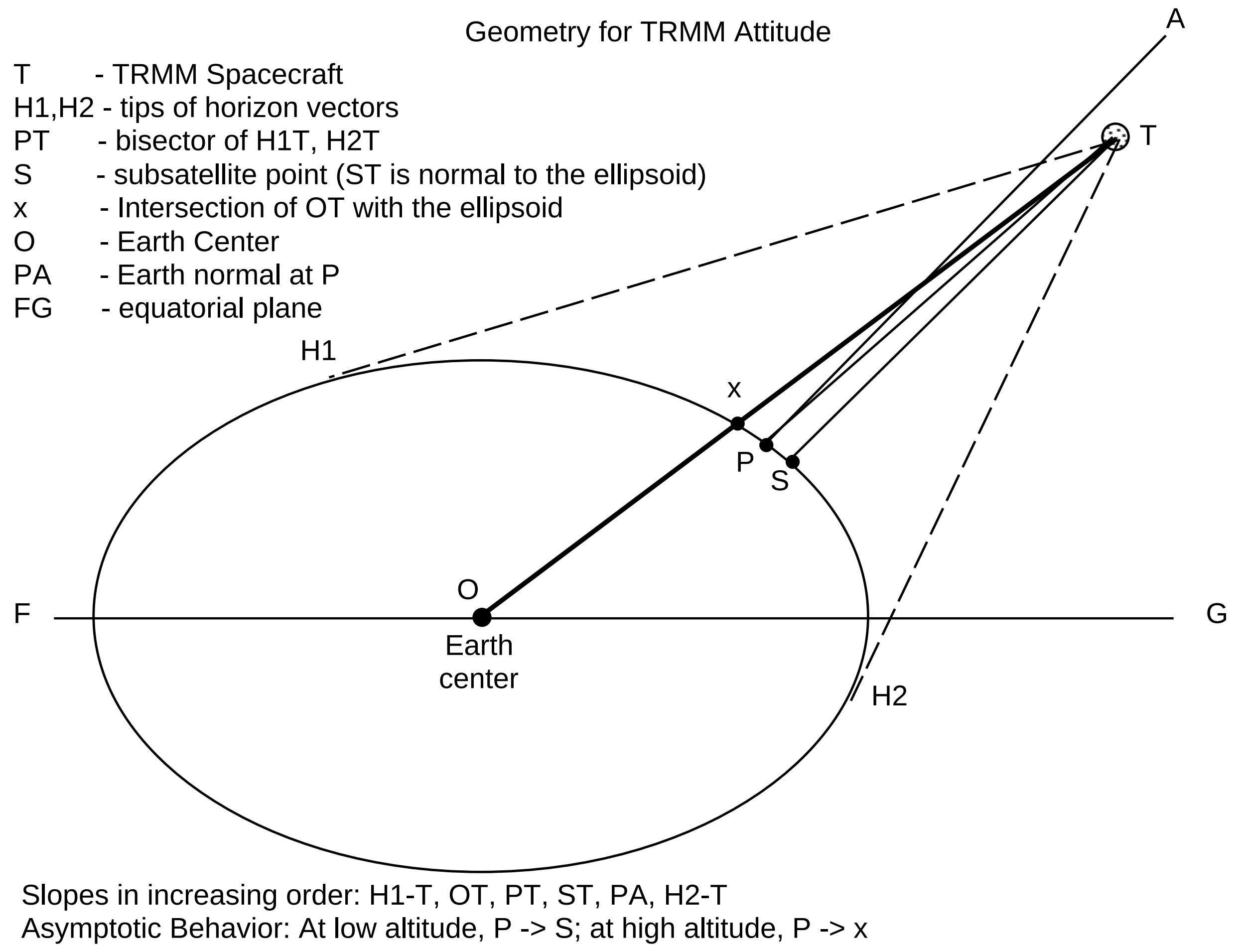

Geometry for TRMM Attitude
T - TRMM Spacecraft
H1,H2 - tips of horizon vectors
PT - bisector of H1T, H2T
S - subsatellite point (ST is normal to the ellipsoid)
x - Intersection of OT with the ellipsoid
O - Earth Center
PA - Earth normal at P
FG - equatorial plane
A
T
H1
x
P
S
O
F
G
Earth center
H2
Slopes in increasing order: H1-T, OT, PT, ST, PA, H2-T
Asymptotic Behavior: At low altitude, P -> S; at high altitude, P -> x

***Figure 3–3. Geometry of the TRMM Tipped Nadir Vector and Other Relevant Vectors***

In Figure 3–3, the true geocentric nadir coordinate system would have its z’ axis along TO. Geodetic nadir is at the subsatellite point S, which cannot easily be determined by using only Earth horizon sensors. Therefore, the TRMM attitude system approximates S by P, which is found by bisecting the lines TH1 and TH2. The line TP is used as the z axis of the “tipped orbital coordinate system.” The yaw angle is calculated from the sun vector, after the tipped orbital system is re–referenced to geocentric nadir by a rotation about an East–West axis by the angle OTP. The approximation that angle OTP = angle OTS is very good for low orbiting spacecraft. Independent validation by EOS shows that it is accurate to better than two arc seconds.

The Toolkit software also establishes the tipped orbital system oriented to P (not S). It is established by a rotation of the (geocentric referenced) orbital system about a suitable horizontal vector as described in the following subsection.

### 3.3.3 The TiltYaw Algorithm

Function:

PGS_CSC_TiltYaw()

This section will consider the transformation needed to handle geodetic nadir referenced spacecraft, with emphasis on TRMM. TRMM obtains its pitch and roll values from Earth horizon sensors, and so it is approximately referenced to geodetic nadir. The methodology will be kept general enough to apply as well to spacecraft in retrograde or polar orbits, although the AM and PM spacecraft are referenced to geocentric nadir, TRMM is prograde, and no polar orbiters are in the program at this time. The TRMM Tipped Orbital system is like the ordinary orbital system except that its z axis is oriented toward approximate geodetic nadir by rotation about a horizontal (East West) line.

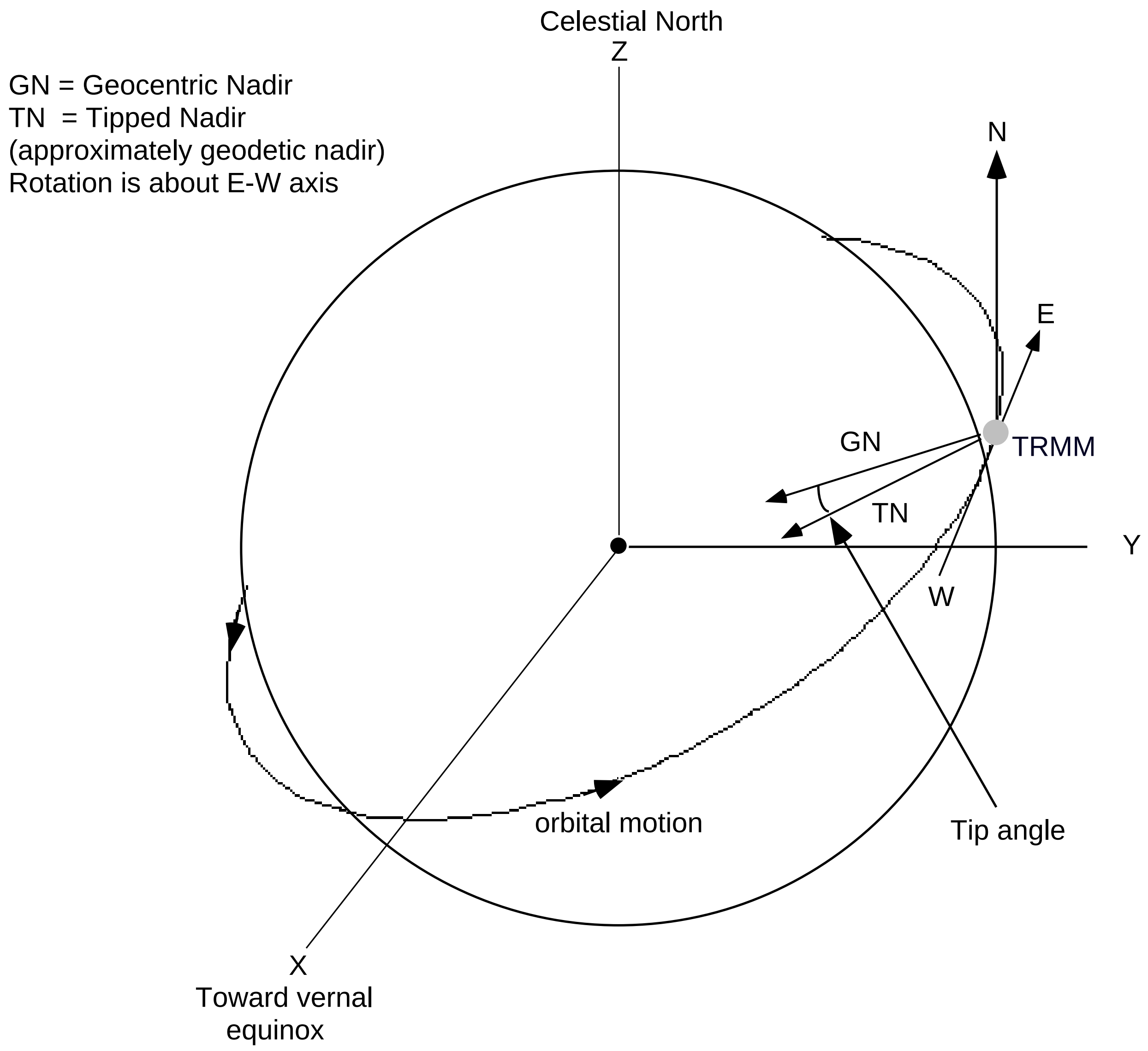


***Figure 3–4. The geometry of the tipped orbital system for TRMM. The "tip" angle is defined about an East West axis "EW***

It is necessary to derive the transformation between orbital and tipped orbital coordinates, so that transformations can be chained: SC to Tipped, Tipped to orbital. This is done by creating a quaternion, quatTIPtoORB, which rotates a vector from tipped orbital to orbital. Since the quaternion rotates vectors, defining their components in one system from those in the other, it must represent a rotation about the same axis as that about which the coordinate axes turn, but the *opposite* way. In other words, if the coordinate system rotates so that nadir turns from approximate geodetic to geocentric, then any vector appears to rotate oppositely–away from the Earth’s equatorial plane. The clearest example is the nadir vector (0,0,1) in Tipped Orbital coordinates–the spacecraft nadir vector when pitch and roll are zero. For simplicity, let us consider the case of a low inclination Eastward traveling (prograde) satellite, so the orbital and tipped orbital y axes are in a nearly North to South direction. In Orbital coordinates, for a spacecraft in the Northern hemisphere, the tipped orbital nadir vector must have a small positive y component. In the rest of this section, I shall always refer to the rotation of *vectors*, not of the coordinate systems.

Let the TRMM spacecraft be at **R** = (X,Y,Z) and have velocity **V** = $(V_x,V_y,V_z)$ in TOD coordinates. The tip angle is (TRMM ASC Algorithm Document Build 3 Version 2, 1994, GSFC)

$$\text{tip} = 2 * f * (A^2/R^2) \sin(\lambda) \cos(\lambda)$$

where f is the Earth flattening factor (A–C)/A. The necessary transformation is a rotation about an East to West axis through the spacecraft in the Northern hemisphere. The negative sign of the tip rotation angle takes care of the difference in the Southern hemisphere, so that an East to West axis is still correct. Let this axis be defined by a unit vector **w**. This axis must horizontal from a terrestrial standpoint, because the rotation must be in the North–South plane **NSP—**the plane through the spacecraft and the Earth’s axis. Let **Z** be the unit vector along the Earth’s axis—T.O.D. celestial North. The rotation axis vector **w** is perpendicular not only to **Z**, but also to the nadir vector **z′**. The reason is that it is perpendicular to **NSP**, and therefore to any vector lying in **NSP**. Since the quaternion quatTIPtoORB must carry vectors from Orbital to tipped Orbital coordinates, its components must be defined in the Orbital system. We therefore seek the three components $w_i$ of **w** in that system.

The condition that **w** must be a vector defined in the orbital system, orthogonal to both **Z**, and **z′** is sufficient to determine it, up to a sign, which must be chosen to make the vector point West. The condition that **w** is perpendicular to **z′** tells us that **w** has the form $(w_0,w_1,0)$. The two numbers $w_0,w_1$ are found from the condition $\mathbf{w} \bullet \mathbf{Z} = 0$. To enforce this condition we need two of the three components of **Z** in Orbital coordinates. These are the dot products

$$Z_0 = \mathbf{Z} \bullet \mathbf{x}'$$

$$Z_1 = \mathbf{Z} \bullet \mathbf{y}'$$

$$Z_2 = \mathbf{Z} \bullet \mathbf{z}'$$

To obtain the first term, we express the unit **x′** vector in orbital coordinates by orthogonalizing **V** to **R** and normalizing the result:

$$\mathbf{x}' = [\mathbf{V} - \mathbf{R}\,(\mathbf{V}\bullet\mathbf{R})/R^2] \,/\, |\, [\mathbf{V} - \mathbf{R}\,(\mathbf{V}\bullet\mathbf{R})/R^2]\, |$$

Only the Z component is actually needed:

$$Z_0 = \mathbf{Z} \bullet \mathbf{x}' = [V_z - Z\,(\mathbf{V}\bullet\mathbf{R})/R^2] \,/\, |\, [\mathbf{V} - \mathbf{R}\,(\mathbf{V}\bullet\mathbf{R})/R^2]\, |$$

The component of **Z** along $\mathbf{z}'$ is not needed per se, but is required in order to obtain the $\mathbf{y}'$ component. It is found as

$Z_2 = \mathbf{Z} \cdot \mathbf{z}' = - Z/R = -\sin(\lambda)$. Finally, $Z_1 = \mathbf{Z} \cdot \mathbf{y}'$ can be calculated by the normalization condition:

$$Z_1 = \mathbf{Z} \cdot \mathbf{y}' = \pm \mathrm{sqrt}(1.0 - Z_0^2 - Z_2^2)$$

The sign is determined by the facts that the $\mathbf{y}'$ axis is opposite to the orbital angular momentum, so it is negative for prograde spacecraft, positive for retrograde. To determine the sense (pro– or retrograde) without reference to the orbital elements, we form the quantity:

$$\mathrm{signEW} = \text{sign of } [V_y * X - V_x * Y] ,$$

which is positive for Eastward travelling spacecraft, negative for Westward. Thus, explicitly, for other than polar orbiters,

$$Z_1 = \mathbf{Z} \cdot \mathbf{y}' = \mathrm{signEW} * \mathrm{sqrt}(1.0 - Z_0^2 - Z_2^2)$$

Finally, the rotation axis vector **w**, which is always Westward, is found from the normalized vector cross product of Z with nadir,

$$w_0 = Z_1 / \mathrm{sqrt}(Z_0^2 + Z_1^2)$$

$$w_1 = -Z_0 / \mathrm{sqrt}(Z_0^2 + Z_1^2)$$

$$w_2 = 0.0$$

The case of polar orbiting spacecraft is special, for signEW = 0. In that case, it is easy to see that **w** is along the - $\mathbf{y}'$ axis when the spacecraft is traveling North, but along $\mathbf{y}'$ when it is traveling South.

Note: The TRMM software is intended to refer the attitude to this system. But, according to our interpretation of the *TRMM ACS Algorithm Document* (GSFC code 712, Aug. 10, 1994), and an additional paper by Tom Flatley, kindly supplied by Dr. Landis Markley, the actual Euler angles generated may not be precisely in reference to that system for two reasons: (1) due to the fact that the geodetic nadir direction does not quite bisect the Earth horizon vectors in a plane containing geodetic nadir. This error is virtually negligible. (2) Potentially larger differences may be generated due to the approximations used in processing the horizon sensor data. For example, in the TRMM Attitude Control System (ACS) algorithms as presented in the *TRMM ACS Algorithm Document*, the matrix "$\mathbf{A}_{\text{Orbit-to-Tip}}$" used to handle the transformation between orbital and Tipped Orbital systems is only approximately orthogonal; thus its inverse cannot be used by the Toolkit for the reverse transformation. While this does not indicate any error or insuperable problem with TRMM attitude data, it means that much care must be taken in later processing of the attitude data. For example, the matrix just referred to is used as part of a sequence of transformations used to generate ECI attitude, which is needed on board only to process the sun vector from ECI to body coordinates, to obtain the yaw. Such considerations indicate that the exact meaning of the TRMM Euler angles must be unraveled by analyzing the TRMM on board and ground data processing algorithms, a task that the Toolkit staff have not yet completed.

The Toolkit 3 release has only a provisional resolution of this problem and completely omits any functionality for handling the different TRMM flying modes. If it turns out that the TRMM Euler angles are re–referenced to zero when the flying mode changes, additional functionality will be added.

In the case of TRMM again, only the spacecraft position and velocity are needed to obtain the quaternions to transform between ECI and orbital coordinates. The spacecraft attitude data are used for the transformation between spacecraft and tipped orbital coordinates. At present, only the ephemeris and an Earth model (Section 6.2.5.1) are needed to transform between the "Tipped Orbital" system and the Orbital System, but after re–analysis of the TRMM on board algorithms it may turn out that the spacecraft attitude data become enmeshed in that transformation. The figure of the Earth is obviously needed in this case, because the spacecraft is essentially referenced to geodetic nadir. The code now uses axes of 6378137.0 and 6356752.31414 meters for the semi–major and semi–minor axes, pending determination of what will be used by TRMM itself.

The transformation between orbital and ECI coordinates is an intimate part of the transformation between ECI and spacecraft. This transformation is handled through the function PGS_CSC_getECItoORBquat(), which returns quaternions from ECI to orbital coordinates based on the spacecraft position and velocity. It was adapted from the UARS simulator. In the case of spacecraft such as AM1 whose yaw axis is referenced to geocentric nadir, then the only task left is the transformation from spacecraft to orbital coordinates

## 3.4 Attitude Interpolation

Function:

PGS_EPH_InterpolateAttitude()

Transformations between Euler angles and quaternions are handled with the methods described in “A Survey of Attitude Representations,” by Malcolm S. Shuster (*Journal of the Astronautical Sciences,* No. **4**, Oct.–Dec. 1993, pp. 439–517). Details are in Section 3.5 below.

To interpolate attitude to time T where T1 < T < T2 we:

a. convert Euler angles at time T1 to an equivalent quaternion, $\mathbf{q}_{T1}$

b. convert Euler angles at time T2 to an equivalent quaternion, $\mathbf{q}_{T2}$

c. Find the quaternion **q** that determines the rotation from the quaternion at T1 to the quaternion at T2, and express it in the form:

$\mathbf{q}_0 = \cos(\theta/2)$

$\mathbf{q}_1 = \mathbf{e}_x \sin(\theta/2)$

$\mathbf{q}_2 = \mathbf{e}_y \sin(\theta/2)$

$\mathbf{q}_3 = \mathbf{e}_z \sin(\theta/2)$

where the quaternion expresses a rotation by angle θ about axis ($\mathbf{e}_x$,$\mathbf{e}_y$,$\mathbf{e}_z$). The algebra for this step is just $\mathbf{q} = \mathbf{q}_{T2}$ (*) $(\mathbf{q}_{T1})^{-1}$, where the (*) operator is quaternion multiplication.

d. Define the quaternion $\mathbf{q}'$ for a rotation from T1 to T by using the same unit vector, ($\mathbf{e}_x$,$\mathbf{e}_y$,$\mathbf{e}_z$) and reducing the angle θ in the ratio (T–T1)/(T2–T1).

e. Find $\mathbf{q}'' = \mathbf{q}'$(*) $\mathbf{q}_{T1}$.

f. Convert $\mathbf{q}''$ to the equivalent Euler angles.

In the case of angular rates we simply do a linear interpolation of the different values (at T1 and T2) to get the values for time T. This is approximate if the reported rates are the rates of change of the Euler angles, and rigorous if they are angular velocity components. The description is as if the numbers are Euler angle rates.

This issue is being checked with the spacecraft attitude teams for TRMM and AM1. The detailed algorithm for the multiplication of quaternions is in Section 3.7

## 3.5 The Toolkit TRMM, EOS AM1 and EOS PM Simulators

The simulators were adapted from a UARS code. The source of the code and documentation are the UARS Programmer Assistance Center at GSFC code 562, Greenbelt, MD (att: Mr. Tom Erickson). The original simulator models the orbits as a Keplerian ellipse expanded to second order in the eccentricity e, with processing node and fixed line of apsides. The latter choice is because the orbits are nearly circular. In Toolkit 3 and Toolkit 4 the mean anomaly increases at the pure Keplerian rate with no J2 correction whatever. This will lead to secular along–track error for the AM, PM, and TRMM spacecraft, although the error was very small for UARS, due to its intermediate inclination. Although the simulator is intended only to enable the user to exercise the software, and not for planning purposes, in Toolkit 5, the mean anomaly rate will be corrected by the factor

$$1.0 + (1.5)J2(A/a)^2(1-e^2)^{-3/2}(1-1.5 \sin^2 i)$$

where A is the Earth’s equatorial radius 6378130.0 m, a the spacecraft’s semi–major axis, and i the orbital inclination. [*Spacecraft Attitude Determination and Control*, Ed. J. R. Wertz (D. Reidel, Holland, 1978), p. 67]. Even after this correction there will remain periodic error in–track of order 50 km, cross–track of order 3.5 km and altitude error ~ 1 km [*The Artificial Satellite Analysis Program (ASAP), version 2.0*, by Johnny H. Kwok (The Jet Propulsion Laboratory, Pasadena, 1987), pp. 2–19 to 2–22]. Thus, users wishing to generate an accurate predicted ephemeris for mission planning should consult FDF at GSFC; the Toolkit simulator is useful only to exercise the software, not for critical planning purposes. The value of the gravitational constant times Earth mass was taken as $3.986005 \times 10^{14}$ m$^3$/sec$^2$, and the oblateness parameter J2 was taken as 0.00108263.

The orbital parameters are:

**_Table 3–1. Orbital Parameters in the Simulator_**

| | TRMM | AM | PM |
|---|---|---|---|
| epoch | 1997–10–01T23:00:00Z | 1998–06–30T10:51:28.32Z | 2000–12–01T10:51:28.32Z |
| semi–major axis (m) | 6729390.0 | 7086930.0 | 7077589.2 |
| eccentricity | 0.00053998 | 0.001281620 | 0.0012 |
| inclination (deg) | 35.0 | 98.199990 | 98.145 |
| longitude asc node[4](deg) | 0.0 | 255.355971130 | 298.54 |
| argument of perigee(deg) | 90.0 | 69.086962170 | 90.0 |
| mean anomaly (deg) | 270.0 | 290.912925280 | 270.0 |

## 3.6 Operations on Quaternions and Euler Angles

This section discusses the transformations between quaternions and Euler angles in Section 3.6.1 and certain ancillary operations in Section 3.6.2. All meaningful rotations of a solid object can be described by rotations using any Euler angle sequence with either three distinct indices, or with the first and last the same, but the middle one different. All combinations are used in spacecraft practice. The Toolkit will denote the three Euler angles phi ($\phi$), theta ($\theta$), and psi ($\psi$) within the code. If the Euler Angle Sequence is (i,j,k) then these angles stand for successive rotations as follows:

$\phi$ represents a rotation about the undisturbed body axis i

$\theta$ represents a rotation about the new body axis j

$\psi$ represents a rotation about the new body axis k

All rotations are defined positive with the right hand rule and are  measured in radians. In this work we assume that always an axis denoted "1" is "X,” "2" is "Y,” and "3" is "Z". Thus, for example, the Euler angle sequence (3,1,2) implies that the angles stand for rotations first about Z, then the new X, then the new Y.

[4]Values of the ascending node, argument of perigee, and mean anomaly are at Epoch

### 3.6.1 Transformations Between Quaternions and Euler Angles

This section describes how to obtain quaternions from the Euler angles.

References:

a. "A Survey of Attitude Representations," by Malcolm D. Shuster, *The Journal of the Astronautical Sciences*, vol **41**, #4, pp. 439–517 (AIAA, Oct. 1993)

b. "Describing an Attitude", by D. I. Kolve, Proceedings of the 16th Annual AAS Guidance and Control Conference, Keystone, CO., Feb. 1993, identical with: *Advances in the Astronautical Sciences* Vol **81**, pp. 289–303 (Univelt, San Diego 1993).

c. *Spacecraft Attitude Determination and Control*, Ed. J. R. Wertz (D. Reidel, Holland, 1978) p. E2, Table E–1

d. "*An Introduction to the Mathematics and Methods of Astrodynamics*", by Richard H. Battin (American Institute of Aeronautics and Astronautics, New York, 1987), Chapter 2.

The meaning of roll, pitch and yaw depends on the order of the angles. Therefore, any tool that uses these variables needs a spacecraft tag to tell it how to transform the angles to other forms. As can be seen in Wertz' book, *Spacecraft Attitude Determination and Control*, p. E2, Table E–1, there are twelve (12) possible orders for the Euler angles. For each spacecraft, we will utilize the correct entries from the Table, but we have set up the transformation to produce quaternions, not the attitude matrix.

For AM1 the representation is 3–1–2, where 1=roll, 2=pitch and 3=yaw (ref: *EOS–AM1 Detailed Mission Requirements*, Draft GSFC Nov. 1993, p. 7110). The nadir reference is geocentric, and the roll axis is negative orbit normal.

For TRMM: It appears from a paper by Tom Flatley (GSFC Code 704), provided by Landis Markley, that the order is 3–2–1 in going from Tipped Orbital to spacecraft, where 1= $\psi$ = yaw, 2=$\theta$= pitch, and 3=$\phi$=roll. There is, however a "tip" angle from geocentric to geodetic and a Mode parameter that has 4 possible values according to whether the +x, +y, -x, or -y axis is forward (more or less along the velocity). Therefore, until we have a full description and analysis of the "tip angle" and the Mode, the TRMM attitude specification is preliminary.

In the following, we use the usual indices 1,2,3 for Euler Angle order. They refer to rotations about the body axes labeled 1,2, and 3 that we shall assume is the same as x,y,z. The angles of rotation are $\phi$, $\theta$, and $\psi$, in order performed. Thus if the Euler angle order is, for example, 2,1,3, it means we rotate by $\phi$ about axis 2, then by $\theta$ about new axis 1, then by $\psi$ about new axis 3.

Let $\alpha$[i][j] = 1.0 i f {i,j} =1,2 or 2,3 or 3,1

= -1.0 i f {i,j} = 2,1 or 3,2 or 1,3

= 0 otherwise

Then if the 3 Euler rotation axes are all different (e.g., 3,1,2), one gets the quaternion from Equation (191) on page 468 of Schuster's paper, but if two axes duplicate (e.g., 3,1,3 which is common) then one must use Equation (192). In the case of duplicate axes, the index "k" is defined as k = 6-i-j, a fact omitted in Schuster's summarization of Kolve's work, but which is fairly obvious, since it is the "missing" index. (E.g., if i and j and 2 and 3 respectively, k is 1, etc.) Because these equations are laborious, consider first the example of AM1 (which uses the order 3,1,2). The quaternion that does not correspond to a space axis is labeled "0". *Incidentally, in Kolve's Equations (3–3) it is necessary to take the indexing on the left hand sides to be that of the Euler angle order, while within the expressions on the right, m', n' and p' stand for the angles $\phi$, $\theta$ and $\psi$ always in standard order; this is corrected in Schuster. Furthermore, Kolve's convention for applying the quaternion to a vector is opposite to that in Battin, Schuster, Shepperd and Dvornychenko; Kolve and Wertz apply the quaternion on the right and its conjugate on the left. We follow Schuster and Battin.* Thus we use

quat[0] = $\cos(\psi/2)\cos(\theta/2)\cos(\phi/2) - \sin(\psi/2)\sin(\theta/2)\sin(\phi/2)$

quat[1] = $\cos(\psi/2)\sin(\theta/2)\cos(\phi/2) - \sin(\psi/2)\cos(\theta/2)\sin(\phi/2)$

quat[2] = $\sin(\psi/2)\cos(\theta/2)\cos(\phi/2) + \cos(\psi/2)\sin(\theta/2)\sin(\phi/2)$

quat[3] = $\cos(\psi/2)\cos(\theta/2)\sin(\phi/2) + \sin(\psi/2)\sin(\theta/2)\cos(\phi/2)$

In the general case, the function should have a setup and an implementation part. The setup should use static variables to index the quaternion and establish any sign conventions needed. Then for the implementation, the angles will change but the integers (or doubles obtained by casting integers). Next I copy the equations from Schuster, and give implementation details.

***Table 3–2. Quaternion components from Schuster's Equation (191) (for angle order i,j,i), (with k = 6-i-j))***

| Quaternion Component Index | Quaternion Component Value |
|---|---|
| i | $\cos(\theta/2)\sin((\phi + \psi)/2)$ |
| j | $\sin(\theta/2)\cos((\phi - \psi)/2)$ |
| k | $\alpha(i,j) \sin(\theta/2)\sin((\phi - \psi)/2)$ |
| 0 | $\cos(\theta/2)\cos((\phi + \psi)/2)$ |

***Table 3–3. Quaternion components from Schuster's Equation (192) (for angle order i,j,k)***

| Quaternion Component Index | Quaternion Component Value |
|---|---|
| i | $\cos(\psi/2)\cos(\theta/2)\sin(\phi/2) + \alpha(i,j)\ \sin(\psi/2)\sin(\theta/2)\cos(\phi/2)$ |
| j | $\cos(\psi/2)\sin(\theta/2)\cos(\phi/2) - \alpha(i,j)\ \sin(\psi/2)\cos(\theta/2)\sin(\phi/2)$ |
| k | $\sin(\psi/2)\cos(\theta/2)\cos(\phi/2) + \alpha(i,j)\ \cos(\psi/2)\sin(\theta/2)\sin(\phi/2)$ |
| 0 | $\cos(\psi/2)\cos(\theta/2)\cos(\phi/2) - \alpha(i,j)\ \sin(\psi/2)\sin(\theta/2)\sin(\phi/2)$ |

In the code, then, a test is first made for duplicate indices; then a branch set up (using a static variable) to Equation (191) in case of a duplicate, or to Equation (192) otherwise. Then the $\alpha$ symbol is created and the indexing of the entries is established using static variables determined from the input setup (in the case of Equation (192) the indices are simply saved; in the case of Equation (191) k must be created from i and j—see below for cyclic permutations.)

Of course, where "$\alpha$(i,j)" is written an ordinary double precision static variable will be used based on code in the Appendix below. Logical branches are set up to distinguish the cases Equations (191) and (192). The logic for setup is:

a. test that none of i,j,k is < 1 or > 3. If there is a violation, return with error for out_of_range_Euler_angle_index

b. see if k is the complement of (i,j) in the set (1,2,3); in other words, test if (i-j)*(i-k)*(j-k) = 0. If not, all 3 indices differ and we have case Equation (192). End of step (B) in this case. If two indices duplicate, we must verify that the third is different, and that it is the middle one! Proceed to step C:

c. Check that i=k and j is not equal to k. If so proceed to use case Equation (191). Otherwise the code returns with error: invalid_Euler_angle_representation

### 3.6.2 How to Create the $\alpha$[i][j] Symbol

The $\alpha$ symbol can be created with simple remaindering arithmetic.

$\alpha$ (i,j) = 1 if j = (i+1) mod 3

$\alpha$ (i,j) = -1 if j = (i+2) mod 3

else $\alpha$ (i,j) = 0

In the C language, a simple prescription for $\alpha$ is

$\alpha$ = (j= =(i+1)%3) - (j= = (i+2)%3)

where "%" is the remaindering operator.

### 3.6.3 Composition of Two Rotations by Quaternions

Function:

PGS_CSC_quatMultiply(quatP,quat,quatout)

The transformations between quaternions and attitude matrices and the way that quaternions act on vectors in the Toolkit are as defined by Shepperd and by Battin (*op cit*). The chaining of quaternion operations was speeded by using Dvornychenko's method.

References:

d. "Quaternion from Rotation Matrix," by Stanley W. Shepperd, *Journal of Guidance and Control*, Vol. **1**, May–June 1978, pp. 223–224.

e. "The Number of Multiplications Required to Chain Coordinate Transformations," by V. N. Dvornychenko, *Journal of Guidance and Control and Dynamics,* Vol. **8**, Jan–Feb 1985, pp. 157–159.

The application of quaternions **q** and **q**' in order results in a third quaternion **q**" whose effect is that of rotating first by **q** and then by **q**'. The multiplication can be done using the representation

$\mathbf{q} = q_0 + \mathbf{i}\, q_1 + \mathbf{j}\, q_2 + \mathbf{k}\, q_3$

and the rule

$\mathbf{i} \times \mathbf{j} = \mathbf{k},$

with its cyclic permutations, and anti–commutative property

$\mathbf{i} \times \mathbf{j} = - \mathbf{j} \times \mathbf{i}.$

This requires 16 multiplications and 12 additions. We follow the more economical method of Dvornychenko, using only 11 multiplications and 19 additions. Because additions are generally much faster, this is estimated to give a factor 1.9 improvement in speed. Dvornychenko's equations will not be reproduced here in original form because then it seems more operations are needed! Instead, we reproduce, in his notation, only the essential definitions and then give additional identities needed to implement the equations in the stated number of operations. In the following, *and here only*, the quaternion for the first rotation is $(q_1,q_2,q_3,q_4)$ and for the second $(q_1',q_2',q_3',q_4')$. The product is **q**"= $(q_1",q_2",q_3"q_4")$. The indexing *here and here only* follows the original author, so that $q_1$ is the component with no space index, while $q_2,q_3$, and $q_4$ correspond to space indices (x,y,z). The arguments must be supplied in the order (q', q). The operations in detail are:

Let

$v_{14} = q_1q_4' + q_4q_1'$

$v_{23} = q_2q_4' - q_3q_1'$

$v_{58} = q_1 q_3' + q_4 q_2'$

$v_{67} = q_2 q_3' - q_3 q_2'$

$q_{24} = q_2 + q_4$

$q_{13} = q_1 + q_3$

$qp_{13} = (q_1' - q_3')$

$qp_{24} = (q_2' + q_4')$

$u_1 = q_{13}(qp_{13} + qp_{24})$

$u_2 = qp_{24}(q_{13} + q_{24})$

$u_3 = qp_{13}(q_{24} - q_{13})$

Then

$q_1'' = u_1 - u_3 + v_{23} + v_{58}$

$q_2'' = u_1 + u_3 - v_{14} + v_{67}$

$q_3'' = v_{58} - v_{23}$

$q_4'' = v_{14} + v_{67}$

This page intentionally left blank.

# 4. Time Streams and Time Transformations

## 4.1 Introduction

This section defines the time standards and time streams for the SDP Toolkit geolocation and data processing; explains the relationships among the different times; and provides or gives references to all the algorithms needed to transform among the times. It describes the data files needed to transform among time streams and the method for updating these files.

### 4.1.1 Organization

This section is organized as follows:

4. 1—Introduction

4. 2—Suggested setup of times for usage in the tools.

4. 3—Time formats

4. 4—Synopsis of the major time streams, their uses and time tools

4. 5—Further discussion; warnings and cautions; day boundaries

4. 6—Detailed description and comparison of the time streams

4. 7—Accuracy issues; tradeoffs in accuracy and storage

4. 8—Relationships between the time streams; algorithms

4. 9—Required data files; maintenance

### 4.1.2 Summary and Overview

This section provides quick summary of the essentials of the time streams. Many time streams and many formats are relevant to processing of EOS instrument data. Later portions of this section go into successively more detail, down to the algorithmic level.

The Toolkit is built around using UTC in ASCII format and Toolkit Internal Time as a double precision number. Toolkit Internal Time, often called secTAI93, is measured in SI seconds from UTC midnight, Jan. 1, 1993. (In Toolkit 4, Toolkit Internal Time was called “TAI.” Strictly speaking TAI is defined from the epoch 1 January 1958, not 1993 and contains other minuscule differences (of order a few nanoseconds) from our internal time. UTC is useful for absolute tagging of data, labeling, and defining base values for processing runs. Internal time secTAI93 is useful for defining time intervals, interpolation, and scientific work. Because it is based on TAI, we shall couch some of the further discussion of it in terms of TAI. TAI is measured in SI seconds, i.e., atomic time as defined by National Institute of Standards and Technology (NIST) and other worldwide bodies. Although TAI may not be as familiar to UTC for many users, the TAI time standard is actually measured by the same atomic clocks that are used to control UTC at the USNO and NIST. Strictly speaking, TAI is measured in seconds and decimals from its starting epoch, Jan 1, 1958. For reasons explained in Section 4.4.4.1. the Toolkit carries TAI instead as secTAI93—seconds and decimals from UTC midnight Jan 1, 1993, and also internally as a Julian Date.

Although the Toolkit offers many time transformations, users normally need be concerned only with UTC, secTAI93, and the spacecraft time for their spacecraft. Many other time streams are available for independent verification, but the necessary ones are used internally in the software. The Toolkit library already contains functions that will obtain the look point, the sub satellite point, the zenith angle and azimuth of the sun, of the moon, and of the look vector (line of sight) at the look point, the slant range and range rate, etc., so that the user need not be concerned with sidereal time, dynamical time (TDT or TDB), Earth aspect, etc. Note that "dynamical time" (TDT or TDB) was formerly called "Ephemeris Time" (ET).

The present discussion is couched within the requirements of EOS instrument data processing. For example, there are additional minuscule differences in time streams that are not reported here, and some of the historical uses of time streams are ignored or slighted. A typical "minuscule" difference is that aside from 19 whole seconds, Global Positioning System (GPS) and TAI times differ by small, empirically determined differences of order < 100 nanoseconds; the differences are available from the USNO. These tiny differences are ignored. Similarly, the way in which the transition was made between use of UT1 and UTC for civil timekeeping is interesting, and it could be significant to users wishing to work with data taken before 1972, but too much space would be required here to discuss possible conversions. [See, however, remarks below about UT1].

## 4.2 Suggested Setup of Times for Usage in the Tools

Tools:

- PGS_TD_UTCtoTAI()
- PGS_TD_TAItoUTC()

Recommended usage is to input all times to the other tools as a UTC starting time plus an array of offsets in SI seconds (TAI seconds). If the original times are in UTC, these offsets can be obtained by using PGS_TD_UTCtoTAI() and differencing. In the case of spacecraft that already have a time stream based on secTAI93 no translation will be needed to get the offsets, but the base time will need to be processed into UTC through PGS_TD_TAItoUTC(). The time offsets can easily be converted (with use of the base time) back to UTC times by using PGS_TD_TAItoUTC(). Note that the UTC seconds field can run as high as 60.9999999... during leap seconds, and users are allowed to input values with seconds fields that large. Users should be prepared to carry this field in this range. On input of ASCII UTC, the software checks that this feature is used only during leap seconds (except in the trivial translation of formats between ASCII A and ASCII B). Any Toolkit function to which is passed an ASCII time with as seconds field exceeding 59.99999... will return without further processing and will issue a diagnostic message, the only exception being simple translation between the ASCII A and B formats.

## 4.3 Time Formats

The word "format" is used in a broad sense, so as to encompass changes in units or in the "base time" or "epoch" from which a time is reckoned, as well as whether the time is expressed in ASCII, floating point, or other formats. The formats used are ASCII; binary coded; and floating point (double–precision numbers or pairs of double precision numbers).

### 4.3.1 ASCII Formats—Details

The Toolkit uses Consultative Committee for Space Data Systems (CCSDS) ASCII formats. These formats are described in the CCSDS Blue Book, Issue 2, *Time Code Formats*, (CCSDS 301.0–B–2) issued by the Consultative Committee for Space Data Systems (NASA Code– OS, NASA, Washington DC 20546), April 1990.

Examples of CCSDS A and B formats are:

CCSDS ASCII A: 1995–03–22T11:22:08.33912Z

CCSDS ASCII B: 1995–081T11:22:08.33912Z

The two times are the same, because Format B uses the day number within the year instead of month and day of month. All tools that accept UTC ASCII times accept either form, but on output of Toolkit calls, form A is always issued.

A detailed explanation of CCSDS ASCII formats used by the Toolkit follows:

#### 4.3.1.1 CCSDS ASCII Time Code A as Implemented by the SDP Toolkit

YYYY–MM–DDThh:mm:ss.d->dZ[ Example 2002–02–23T11:04:57.987654Z ]

where

YYYY = a four character subfield for year, with value in range 0001–9999

MM = a two character subfield for month with values 01–12, leading zeros required

DD = a two character subfield for day with values in the range 01–eom, where eom is 28, 29, 30, or 31 according to the month (and, for February, the year)

The "T,” a separator, must follow the DD subfield; if and only if there are more characters after the DD subfield; the string will be accepted and parsed such that mm, ss, and d are treated as 0. In that case, a "Z" will still be accepted, but not required, at the end.

hh = a two character subfield for hours, with values 00–23

mm = a two character subfield for minutes, with values 00–59

ss = a two character subfield for seconds, with values 00–59 (00–60 in a positive leap second interval, 00–58 in the case of a negative leap second)

d->d an n–character subfield, (n < 7 for input n = 6 for output), for decimal fraction of a second, with each digit in the range 0–9. If the decimal point appears on input, digits must follow it.

Z - terminator, optional on input

NOTE: The CCSDS Formats require all leading zeros be present.

#### 4.3.1.2 CCSDS ASCII Time Code B as Implemented by the SDP Toolkit

The CCSDS ASCII Time Code B format, described on p. 2–7 of the Blue Book, is:

YYYY–DDDThh:mm:ss.d–>dZ

[ Example 2002–054T11:04:57.987654Z ]

The format is identical to the Code A except that the month, day combination MM–DD is replaced by day of year, i.e.,:

DDD = Day of Year as a 3 character subfield with values 001–365 in non leap years and 001–366 in leap years.

NOTE: The CCSDS Formats require all leading zeros be present.

The B format will be processed similarly to the A; for example, a fatal message will issue if DDD = 366 in a non leap year, just as a fatal message would have issued in Code A for month = 02 and day = 29 in a non leap year.

The output strings will be 27 characters in Code A, including the "Z,” and 25 in Code B, including the "Z".

### 4.3.2 Real Number Formats—Details

The floating point formats for times are:

- double precision floating point offsets and differences in SI seconds
- two double precision floating point numbers ("double double") for Julian Dates (JD)
- seconds from 1993 Jan. 1, UTC midnight, as a double precision number
- Julian Days—always reckoned from Greenwich Mean Solar Noon, Jan. 1, 4713 BC. The terms Julian Date and Julian Day Format are used interchangeably here. Julian Day number, used elsewhere, is simply the integer part of our Julian Date. The "modified Julian Date" is used only internally, to access certain tables. It is defined by: MJD = JD - 2,400,000.5. The situation is discussed pictorially with examples in Appendix B. Toolkit names for time streams contain the sequences "jd" or "jed" if in Julian day format. The notation "jed" is used in TDT and TDB because they correspond to the older Ephemeris Time and can be used in software that calls for "Julian Ephemeris Date," or "Julian Ephemeris Days."
- double precision values in radians—used for Greenwich Mean and Apparent Sidereal times, which are measures of Earth rotation

### 4.3.3 Julian Dates—Details

All Julian Date formats are two double precision numbers as an array. On output, the first is always half–integral and the second lies between 0 and 1. Users wishing to carry a Julian Date as one double precision number *MUST* add the two parts. In Toolkit 5, there will be tools that accept Julian Day input. To use these tools, on input, for optimum precision, the user should supply two doubles parsed as indicated; however it is possible to set the first member to the total Julian Date and the second member zero with some loss of precision. In this case the accuracy will be about 2 milliseconds (for a computer that maintains double precision numbers to 14 decimals accuracy), while with the "double doubles" the accuracy will be better than a microsecond—it is essentially limited by timing errors and not by roundoff. (Note: 14 decimals is probably a worst case; several UNIX test platforms at the Toolkit development site have performed to 15 decimal figure or better accuracy.)

Example:

| | | |
|---|---|---|
| UTC time 1996–03–31T00:00:00Z | -> | TDB[0] = 2450173.5000 |
| | | TDB[1] = 0.00070816731 |

## 4.4 Synopsis of the Major Time Streams, Their Uses, and Time Tools

This section gives further detail on the time streams in a succinct manner; refer to Section 4.5 for additional description.

### 4.4.1 Long Term Time Streams (spanning many days or years)

This section summarizes the tools dealing with long term time streams

Tools:

- PGS_TD_UTCtoTAI()
- PGS_TD_UTCtoGPS()
- PGS_TD_UTCtoTDTjed()
- PGS_TD_UTCtoTDBjed()

Functions:

- PGS_TD_UTCtoTAIjd()
- PGS_TD_UTCtoUT1jd()
- PGS_TD_UTCtoUTCjd()
- PGS_TD_UTCtoTDTjed()
- PGS_TD_UTCtoTDBjed()

Where

– UTC—used for civil timekeeping. Zone times (Eastern Standard, etc.) are obtained by adding [subtracting] whole hours (or, in rare cases, half hours) to UTC, approximately 1 hour for each 15 degrees of longitude East [West], but the exact boundaries of the zones are established politically. The boundaries tend to run straight North to South on 7.5 degree centers in the oceans, the denizens thereof apparently being politically indifferent to such matters. There is a map of time zones in the *1992 Explanatory Supplement to the Astronomical Almanac* on p. 57. The Toolkit needs UTC as a Julian Day or Modified Julian Day, internally, to access the files of UT1–UTC, polar motion, and leap seconds, because the USNO and other agencies have set up the files in this way.

– TAI—used for accurate timing of any physical phenomena in the near Earth environment. This is the time that scientists use (in principle) when employing a well–calibrated laboratory clock, i.e., it is the time by which they ought to calibrate their clocks. In the Toolkit it is represented by secTAI93.

– GPS—SI seconds from Jan 6, 1980. In Toolkit 4 GPS was represented as TAI minus 19 seconds, according to certain USNO bulletins, but in Toolkit 5 it will be re–referenced to Jan 6, 1980. See Section 4.4.4.2 for details.

– TDT—used for accurate timing of orbital phenomena in the near Earth environment Equal to TAI + 32.184 seconds. Provided as a Julian Date.

– TDB—used for accurate timing of orbital phenomena in the solar system. Used to access the Solar/Lunar/Planetary ephemeris. Provided as a Julian Date.

– UT1 as a Julian Date—used to measure Earth rotation.

Note: There are various Spacecraft (SC) times used. So far it is known at the time of this writing; AM1 will provide UTC. TRMM originally stated that it would provide continuous seconds from Jan. 1, 1993, but now appears to be offering spacecraft clock time since power–up plus additional files and file header information that may suffice to recover the originally stated time stream. [TRMM Science Data and Information System (TSDIS), private communication] It is hoped that a workable method will be found to obtain seconds from Jan 1, 1993 or UTC from the TRMM clock; the subject is under negotiation.

### 4.4.2 Tools for Daily Repeating Term Time Streams (returning to zero in approximately one day)

This section summarizes the tools dealing with daily repeating times. Several are provided to high precision (roughly better than 0.01 sec—see Section 4.7 for details), while others are only approximate (good to a few seconds). Users wanting accurate sun angle should use the precision Solar ephemeris and the Zenith–Azimuth tool rather than the approximate times, which are useful mainly for quick checks.

#### 4.4.2.1 Precision Times

Tools:

- PGS_TD_GMST() (Toolkit 5)
- PGS_TD_GAST() (Toolkit 5)
- PGS_TD_UTCtoUT1()

Where

- GMST (Greenwich Mean Sidereal Time)—the hour angle of the mean vernal equinox of date at the Greenwich Meridian. (The mean equinox is affected by precession but not by nutation.) This angle increases more uniformly than Greenwich Apparent Sidereal Time (GAST) because it is independent of nutation, but it cannot be used directly to measure true Earth orientation. Provided in the Toolkit in radians.
- GAST (Greenwich Apparent Sidereal Time)—the hour angle of the (true) vernal equinox (of date) at the Greenwich meridian. This angle is used to obtain the Earth rotation angle in PGS_CSC_ECItoECR() and PGS_CSC_ECRtoECI() because of the way UT1 is measured–relative to the true equinox of date. Provided in the Toolkit (Toolkit 5) in radians. (Note: The IERS Standards calls GAST “GST”)
- UT1 as seconds from midnight—usable as a measure of Earth rotation. It is really just GAST reduced to a solar day basis instead of a sidereal day basis. Caution is needed around midnight in using this time. Internally, the Toolkit uses a Julian Date form of UT1 that does not jump at midnight.

#### 4.4.2.2 Approximate Times

Tool:

- PGS_CBP_SolarTimeCoords()

The following are provided only approximately (maximum error about 6 seconds):

- Greenwich Mean Solar Time—time at Greenwich based on the hypothetical motion of the "mean sun,” which traverses the celestial equator (not the ecliptic) once a year at a uniform rate. (The true sun moves at a nearly constant rate on the ecliptic.) This time, called UT0 was, historically corrected for polar motion and used as the basis for UT1—a fundamental measure of Earth rotation. Nowadays, UT1 is based on other measures of Earth rotation, such as Laser Geodynamics Satellite (LAGEOS), and Very Long Baseline Interferometry (VLBI). Thus Greenwich Mean Solar Time is not currently of much importance, but could be useful for interpreting historical data.
- Greenwich Apparent Solar Time—time at Greenwich based on the angle of the true sun from the Greenwich meridian, allowing for variations in its motion due to the inclination of the Earth's axis and the eccentricity of the Earth's orbit, and aberration. This time is what one would measure at the Greenwich meridian with a good sun dial. The difference

in Apparent and Mean Solar time is called the "Equation of Time." It is exhibited as a graph on p. 6 and as the "analemmic curve" on p. 486 of the 1992 *Explanatory Supplement to the Astronomical Almanac*, and an expression for it is in Equation (9.311–3) on the same page, which reads:

GAST–GMST = -1.915 sin(G) - 0.020 sin(2G) + 2.466 sin(2$\lambda$) degrees, where G = 357.528 + 35999.050 T degrees, and T is the UT1 time from J2000 in Julian Centuries.

Although Apparent Solar time can be used to obtain an estimate of the sun angle, the user is advised to use the tools PGS_CBP_Earth_CB_Vector() to get the ECI sun vector, PGS_CSC_ECItoECR() to put the vector into ECR, and the tool PGS_CSC_ZenithAzimuth() for an accurate sun zenith angle and azimuth.

- Local Mean Solar Time and Local Apparent Solar Time—these are the same as at Greenwich, but adjusted for longitude at the rate of one hour time per 15 degrees in the obvious way (clock reads earlier as one travels West, later as one travels East). Local mean Solar time will generally be within about 1/2 hour or so of local zone civil time, depending on how time zones have been drawn. Local apparent solar time will be what one would measure locally with a well–calibrated sun dial.

### 4.4.3 Recommended Usage of the Time Streams

The following table summarizes the recommended usage of the time streams.

***Table 4–1. Recommended Usage of Times in EOSDIS Data Processing***

| Time Stream | Recommended Usage |
|---|---|
| UTC (ASCII) | Labeling data batches, files, figures<br>Initializing Toolkit software runs (when combined with offsets)<br>Comparison with foreign data |
| UTC (Julian Date) | accessing tables of leap seconds, polar motion<br>Rough correlation with foreign data (up to 1 second error possible)<br>NEVER for time differencing!!! |
| TAI | Taking time differences; interpolation, smoothing, differentiation<br>Offsets to be used as inputs to Toolkit functions<br>Orbital mechanics and any scientific algorithms around the Earth (radiative transfer, fluid flow, mass transfer, geochemistry, etc.) |
| GPS | Ground Truth comparisons with GPS based data |
| Spacecraft Time | accessing spacecraft data (Ephemeris and attitude data are accessed with UTC and offsets; spacecraft time would be needed only for science data.) |
| The following times are used internally and will be provided as a convenience to users, but are not expected to be needed in practice | |
| TDT | orbital mechanics about the Earth |
| TDB | Solar System orbital mechanics; lunar orbital mechanics |
| UT1 (Julian Day) | could be used as input for user's Sidereal Time software (Toolkit 5) |
| UT1 (seconds) | Earth orientation |
| GMST | Earth orientation |
| GAST | Earth orientation (Toolkit 5) |

### 4.4.4 Starting or Base Epochs and Minor Differences

Various time streams start at different times. Other than for Julian days, no effort has been made to base times on the official start.

#### 4.4.4.1 TAI Start Epoch

We have used TAI as a Julian Date and as seconds from 1993 Jan 1, but not as seconds from Jan 1, 1958, the official epoch. See Section 4.1.2. Note, that as a Julian Date, it will differ from the UTC Julian Date by the leap seconds (divided by 86,400), exactly. There is no pretense that UTC nor TAI as a Julian Date correctly counts Earth rotations from the Julian Date epoch 4713 BC. In fact, UT1 would be closer to a correct count.

#### 4.4.4.2 GPS Start Epoch

The official time origin for GPS , as defined by the United States Air Force (USAF) Falcon Air Force Base is UTC midnight, Jan. 6, 1980. Based on information from the USNO Time Service (e.g., Announcement Series 14, No. 56 of Feb. 1, 1994) Toolkit 4 calculated GPS as TAI minus 19 seconds; but the 19 second difference is the number of leap seconds on Jan. 6, 1980, and the Toolkit TAI is from Jan. 1, 1993. To prevent confusion, in Toolkit 5 GPS will be seconds from the official start epoch, Jan. 6, 1980. (There were 409881608 seconds from Jan. 6, 1980, 00:00:00 UTC to Jan. 1, 1993, 00:00:00 UTC, but the new GPS will exceed the Toolkit 4 value by 409881627 s.) The truncation error will deteriorate from < 1 microsecond to several microseconds in Toolkit 5 as GPS will now keep track of the large constant difference from 1980 to 1993.

#### 4.4.4.3 Modified Julian Dates (MJD)

The modified Julian Date MJD is defined as the Julian Date (JD) minus 2400000.5. Thus it can be considered as based on an epoch of Nov. 17, 1858. The Toolkit does not explicitly offer the MJD for reasons that are explained in Sections 4.5.4 and Appendix B, but it uses them internally to access the tables of leap second, polar motion, and UT1–UTC. The MJD may be offered in Toolkit 5.

#### 4.4.4.4 Minor Differences

The U.S. Naval Observatory bulletin boards keep track of minuscule fractional second differences between UTC, TAI and GPS separately from whole second differences. These differences exist because, although UTC is kept as close to (TAI—leap seconds) as feasible, in fact TAI is available only about 90 days late. Therefore, various estimates are used to set UTC in the interim, and, in the final analysis, small differences will exist once TAI is made definitive. To EOSDIS accuracy requirements, these errors of order < 100 nanoseconds are inconsequential, and we freely assume that UTC and TAI differ only by whole seconds. Similarly, NIST and USNO may differ by amounts measured in nanoseconds in their evaluation of UTC. Both depend on the Bureau International de l’Heure (BIH) for TAI.

## 4.5 Further Discussion; Warnings and Cautions; Day Boundaries

Many of the available time streams run at nonconstant rates (in terms of the SI second) or depend on data files for their maintenance.

### 4.5.1 Which Time Streams Run Consistently With Atomic Time?

Note that UT1, both kinds of sidereal time, and all the varieties of solar time run at varying rates and cannot be used for meaningful time differencing (nor can UTC). TAI or TDT can be so used. Greenwich Mean Solar Time runs sufficiently constantly that it could, however, be used for differencing. In data before 1972 Greenwich Mean Solar Time may well be the time that was used. No conversions are provided from either Sidereal time or from UT1 to any of the group (UTC, TAI, TDT, TDB). The reason is that such conversions would be of interest only to Earth based optical observers doing time determinations by observing, e.g., the sun or another star. Again, we repeat, that although UTC runs consistently with atomic time "most of the time" it does *not* do so at leap seconds.

### 4.5.2 Problems With UTC as a Real Variable

"Do NOT attempt to use jdUTC for any differencing, interpolation, etc., Use TAI or TDT"

UTC is used as a basis for most Toolkit work, because it can readily be related to civil time, and the offsets to other time streams are defined from it. UTC is virtually always expressed in ASCII. Occasionally, UTC may be expressed in Julian Day format, but in this case it is very deceiving and *great caution* must be used at leap seconds. The reason it is offered at all in Julian Day format is that it has to be used as a Julian Day to access the table “*leapsec.dat*” of leap seconds, and as a Modified Julian Day to access the table “*utcpole.dat*” for polar motion, and UT1–UTC time differences. The EOSDIS planetary ephemeris, originally obtained from JPL, is accessed with TDB as a Julian Date, which is obtained through a series of steps from UTC as a Julian Date. (See TDB). For the AM1 spacecraft, UTC is coded in CCSDS Day Segmented (CDS) format (see the User Guide).

The problem with UTC as a real number is not entirely confined to leap second intervals, because there is a cumulative effect of all the leap seconds. That is because, in the long term, UTC tracks UT1, and the UT1 second is not an SI second. It is closer to a mean solar second. Thus, over a long interval, time differences in UTC Julian Days, for example, are not compatible with time differences in SI seconds.

### 4.5.3 Julian Dates

Users are urged to read the cautions in regards to Julian Days. The use of Julian days and Julian Dates is very convenient for keeping a long term continuous record, but the use of Julian Days/Julian Date is *not* a way to avoid interchange among time streams; each runs at its own rate or offset, so that a Julian Date in UTC will *not* in general coincide with a Julian Date in UT1, TAI, or anything else. The Julian Date is most useful for providing interchageability among different data sets. The Toolkit converts whole Julian day numbers to and from Gregorian dates with functions from the U.S. Naval Observatory (Appendix B). The fractional parts and day boundaries are handled using the necessary offsets as described in the following section and in Appendix B. Although Julian Dates in UT or UT1 may originally have been intended to count Earth rotations from the original epoch, 4713 BC, the significance of Julian Dates for EOSDIS/SDP lies in the conversions to UTC, TDT, etc.

### 4.5.4 Day Boundaries in UTC , Julian and Modified Julian Dates

Julian Dates always start at noon, in the following sense: The very beginning is set to noon, Greenwich mean solar time, 4713 BC. Thereafter, to this day, each Julian day begins near the next UT1 noon, but the various noons, such as UT1, TDT, or UTC, may drift from each other, because the different time streams have different physical bases. [Note that if UT1 is expressed as a Julian Date this implies the antinomy that when the Julian Date of UT1 is a whole number, the UT1 time is noon, but when UT1 is expressed instead as a difference from UTC, it is always a fraction of a second.] As time progresses, Julian days will "roll over" at various times that are not trivial to predict.

It is quite fortunate that the rollover of ordinary Julian Dates is well separated from the midnight rollover of other time formats, because it will be thousands of years before any discrepancy causes problems in matching Julian dates with other dates. In other words, if the drifts implicit in the introduction of leap seconds continue, it may happen someday that (for example) TDB beginning of day, now near UTC noon, moves near UTC midnight, so that TDB rolls over its whole day number at almost the same time as UTC. The modified Julian day suffers from this problem right now; for it rolls over at midnight. Great caution should therefore be used in working with MJD whole number values when the fraction is small; the whole day number for various time streams as Modified Julian Days will change at times near, but not generally quite at, UTC midnight. The Toolkit tables and functions that access the tables are set up to work reliably in these regions. See Appendix B for examples of how the rollovers in UTC and Julian Days interlace, while the Modified Julian Day changes nearly at the UTC change of day (exactly on it if the MJD refers to UTC).

### 4.5.5 All Sidereal Times and UT1 Depend on Frequent Updates of Data Files

It is important to understand that sidereal time (either mean or apparent) is set up as a measure of true axial Earth rotation. Therefore, it is directly tied to UT1, and cannot be determined a priori; its knowledge requires the tables of UT1–UTC, ("*utcpole.dat*") which are available only post hoc. The reason is that sidereal time is used in astronomical observation (optical and radio) and so it must reflect the true attitude of the Earth. Older software and literature often purport to find sidereal time based a some simple algorithm. Generally such methods have a substantial error penalty, of the order of more than a second, possibly several. Such algorithms useful as crude checks only. Similarly, some references consider GMST to refer to the inertial location of the Greenwich meridian. But the Greenwich Meridian is an hour circle (semi–circle centered on Earth and passing through the North and South poles) whose inertial location cannot be specified by one number; rigorously, one needs to know the pole location and an angle of rotation about it. GMST actually refers to the angle measured from the Greenwich meridian to the mean equinox of date; being the angle from a curve to a point, it is uniquely defined.

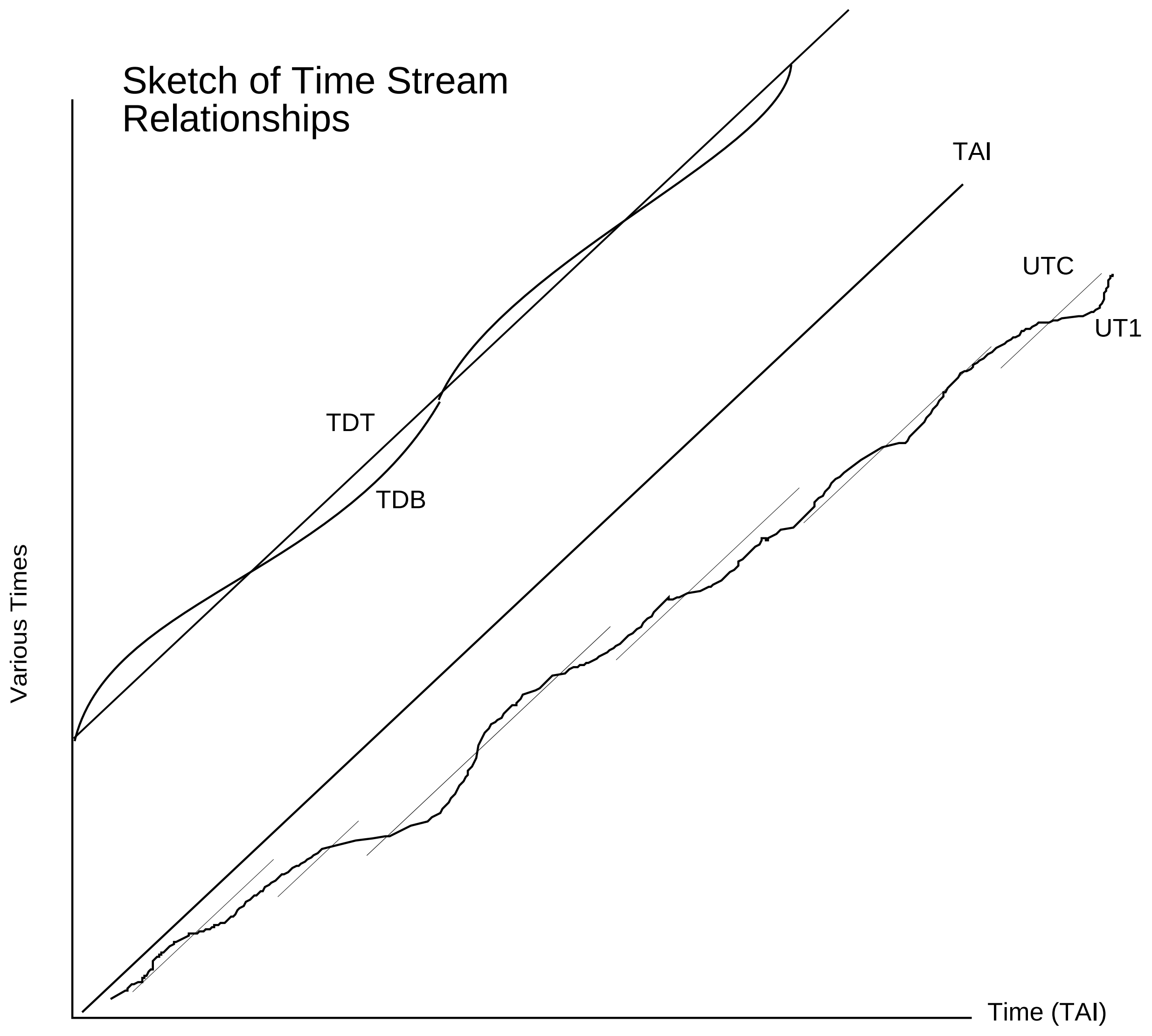
It is remarkable that the Astronomical Almanac appears to offer (for example in the 1995 Almanac in Table B10) sidereal time more than a year in advance, as a function of Universal Time. Of course, on close reading (p. L2, line 29), one sees that the Universal Time to be entered in the tables is UT1, which can be found from UTC only after the measurements of Earth rotation are available (e.g., from the USNO or IERS). Thus, the value of UT1 is needed to use the tables, although one can get within ~0.9 sec by using UTC.

## 4.6 Detailed Description and Comparison of Time Streams

This section inter–relates the various time streams in depth, in preparation for presenting the algorithms. Section 4.6.1 gives an overview, 4.6.2 details, and 4.6.3 further details on UT1, which deserves extra attention because it relates so closely to Earth rotation.

### 4.6.1 Overview of the Long term Behavior

The accompanying figure shows schematically the relationships among UTC, UT1, TAI, TDT and TDB. (Sidereal time being related to UT1 is omitted.) The figure is by no means to scale; the jumps at the leap seconds are exaggerated, as are the 32.184 second difference between TDT and TAI, and the periodic variation of TDB. Note that UT1 falls more and more behind TAI and TDT, as time progresses. The fluctuations of UT1 are exaggerated, too; UTC–UTC has a sawtooth form that is simply explained: The UT1 second is longer than the UTC second, which causes it to lag, but then UTC is dropped downwards at a leap second, to compensate. On top of the resulting sawtooth are seasonal variations and short term fluctuations, but the sawtooth is clearly discernible. (TAI in the figure is offset by the leap seconds only, not taking into account any difference in epoch, i.e.; it is Toolkit internal time.)



Only a section of TDB is shown. The downward jumps in UTC (at leap seconds) occur in response to the slow downward drift of UT1 and ought properly to lag the deficiency of UT1 relative to UTC.

***Figure 4–1. Sketch of Time Stream Relationships***

There are two items of concern to the Toolkit regarding this increasing divergence:

a. It is desirable to have a short term approximation to the function UT1–UTC to handle cases of gaps in the importation and processing of Earth motion data.

b. It is desirable to have some conception of how many leap seconds to expect in the next few decades, and if any negative leap seconds can be expected. The long term behavior of UT1 will be discussed with these points in mind.

### 4.6.2 Details on the Individual Time Streams

This section explains the differences among the time streams in more detail.

The most important considerations are:

a. UTC runs at a constant rate in SI seconds, except at leap seconds, where it is discontinuous. Leap seconds always occur at the end of the 24 hour day, when the hours field is 23, the minutes 59, and the seconds are at 59.99999..... (as many 9's as decimals are kept). During the (normal, positive) leap second, the seconds field is allowed to run past 60.0000 up to 60.99999 seconds, and then it is reset to zero as the day is incremented and the hour and minute counters are reset from 24 and 59, respectively, to 0. If all the fields are combined into one real number, that number will jump backwards by one second at the end of the leap second. Therefore, UTC as a real number cannot be used to label data uniquely. For this reason, the Toolkit maintains UTC generally as an ASCII string. In a few cases, UTC is temporarily, internally converted to a Julian Date, after which immediately, the leap seconds are added, to make a monotonically increasing time stream. In a very few rare cases, UTC could be used as a Julian Date without concern for the leap seconds, but the effect on different determinations will be quite different. For example, Earth rotation can be inaccurate up to ~ 450 m due to a ~ 0.9 sec error (the error in UT1 that could accrue from using UTC instead), while the larger error of up to ~30 omitted leap seconds in accessing the planetary ephemeris with UTC instead of TDB would result in a smaller error of planetary motion, (except possibly for the moon). The error in the position of the moon could be substantial. A negative leap second has never occurred, but if it did the seconds field would then run only to 58.999999... after which it would return to 0.00000 with the minute and hour fields as the day is incremented. In that case, UTC as a real number would jump one second forward. In summary, it can be seen that UTC as a real number (e.g., a Julian Day) is unsuitable for use in scientific algorithms, or even for interpolation, because it is discontinuous, and, furthermore, in the case of a positive leap second, it cannot be used to label data uniquely if it is converted from coded form (ASCII or segmented binary) to floating point form. All these problems arise because UTC is slaved to stay within about 0.9 seconds of UT1, and to run at the same rate as TAI in between leap seconds, while UT1 is determined empirically from Earth rotation. There can never be a time stream that runs at a constant rate and agrees with Earth rotation, because the latter is variable. Thus, one is always stuck with times that don't track civil time very well, or times that run at irregular rates.

b. TAI, TDT and TDB are all continuously increasing time streams readily accessible or readily computed from UTC or other equivalent data. TDB is used mainly for planetary ephemeris work, TDT for ephemeris work on Earth–orbiting spacecraft, and TAI for laboratory use. GPS is TAI – 19 seconds (the number of leap seconds as of Jan 6, 1980, which is the origin epoch for GPS). TAI and TDT run at the same rate exactly. TDB is "corrected" by periodic terms of order up to 1.6 milliseconds for relativistic effects, so as to run at a constant rate if observed at a large distance from the sun. The corrections are due to variations in the sun's gravity potential and the Earth's orbital velocity, because the

Earth's orbit is eccentric. Of course, in a Barycentric reference system, it is TDB that runs at a constant rate and the other time streams that have small periodic variations.

c. UT1 is no longer used for timekeeping; it is used as a measure of the rotation of the Earth. In particular, the difference UT1–UTC is the fundamental datum, obtained from the US Naval Observatory or the IERS in Paris, which determines the exact axial rotation of the Earth as a deviation from constant rotation. UT1 now runs slower than TAI, on the average, but UTC runs exactly at the rate of TAI in between leap seconds. See Section 4.3 for details and estimates.

d. Spacecraft time will come to us in different forms for different spacecraft. For AM1, it will come in UTC in CCSDS Segmented Library format. For TRMM, it was originally specified to come in continuous seconds since Jan 1, 1993, in CCSDS Unsegmented Time Code (CUC), but it has recently been altered so that the leap seconds are removed. This means that TRMM clock time, as reported to Earth Science Data and Information System (ESDIS), will jump backwards one second at the conclusion of each leap second. It is hoped that TRMM will supply auxiliary data that can be used to restore the originally promised time stream.

e. The difference between Apparent and Mean Sidereal times is called the "Equation of The Equinoxes.” It is exhibited in tabular form in the *1995 Astronomical Almanac* on pp. B8–B15. The algorithm is given later in this document. In simple terms, the reason that the apparent time must be used is that UT1 is the measured quantity that must be used to get accurate Earth rotation, and it is based on apparent sidereal time, partly for historical reasons. The mean sidereal time is the one that can more readily be calculated from a simple equation, not involving nutation.

### 4.6.3 Explanation and Estimate of the Secular Variation in UT1

At present, there are about 0.8 leap seconds a year, which implies that the UT1 day is, on the average, about (86400.8/86400.0) TAI or TDT days. This estimate is based on a time base from 1958, when TDT–UT1 was 32.18 sec, to Jan 1, 2000, when the U.S. Naval Observatory estimates (in Series 77 data) that TDT–UT1 will be 65.824 sec, with a probable error of 3.16 sec. According to K. L. Lambeck, *The Earth's Variable Rotation: Geophysical Causes and Consequences*; (Cambridge University Press, 1980), the day is slowly lengthening at the rate of about 0.0015 sec/cy, due to the tidal effect of the moon and sun; it is also affected by motions in the Earth's core, changes in the polar ice caps, and other geophysical effects. These are discussed in Lambeck, op cit, or *The Earth's Rotation from Eons to Days*, Ed. P. Brosche and J. Sundermann (Springer–Verlag, Berlin and New York, 1988). On the basis of recent behavior, the rate at which leap seconds will be added ought to increase over the present rate of 0.8/year only by about 0.55 additional leap seconds per year per century. Thus, for EOSDIS purposes, we can assume that there will be no gross increase in the rate of accumulation of leap seconds; furthermore, with the passage of time it becomes less and less likely that one will encounter a negative leap second. It is impossible to take a long term view of the evolution of Earth rotation by examining the accumulation of leap seconds, because they have not been used for a long enough period of time. An approximate idea of the development of Earth rotation can be gleaned, however, from a long term comparison of ET and UT1, where ET is "Ephemeris Time" (now best identified with TDB). UT1 can be estimated in the distant past from historical observations of the sun, eclipses, etc., and ET can be assumed to run at the same rate as TAI indefinitely in the past. Currently (late 1994) the difference is about 61.184 seconds (TDT–TAI = 32.184s plus 29 leap seconds). The rate of accumulation of leap seconds is found from the rate of increase of the length of day by a units change:

- l.o.d. change ~ 0.0015 s/(day–cy) = 0.0015 s/(day–cy) * (365.25 days/y) ~ 0.55 s /(y–cy)

Long term records are kept not in terms of length of the day (l.o.d.), but as time differences. If the foregoing law is integrated, one can obtain a quadratic relationship of delta(T) versus time, where delta(T) = ET–UT1.

The tables on pp. K8–K9 of the 1995 Astronomical Almanac (augmented with predictions from USNO Series 77) were therefore fitted with a quadratic function, but with the constants adjusted by eye to give a good fit in the late 1980's to 1994. The result is:

- ET–UT1 ~ (1/2) (0.537 s/cy) [year–1844]$^2$/100 (years/cy)

The factor of (1/2) from integration is kept explicit so that the rate of increase of leap seconds per century, namely 0.537, can be read off the equation. The function is plotted with the data in Figure 4–2.

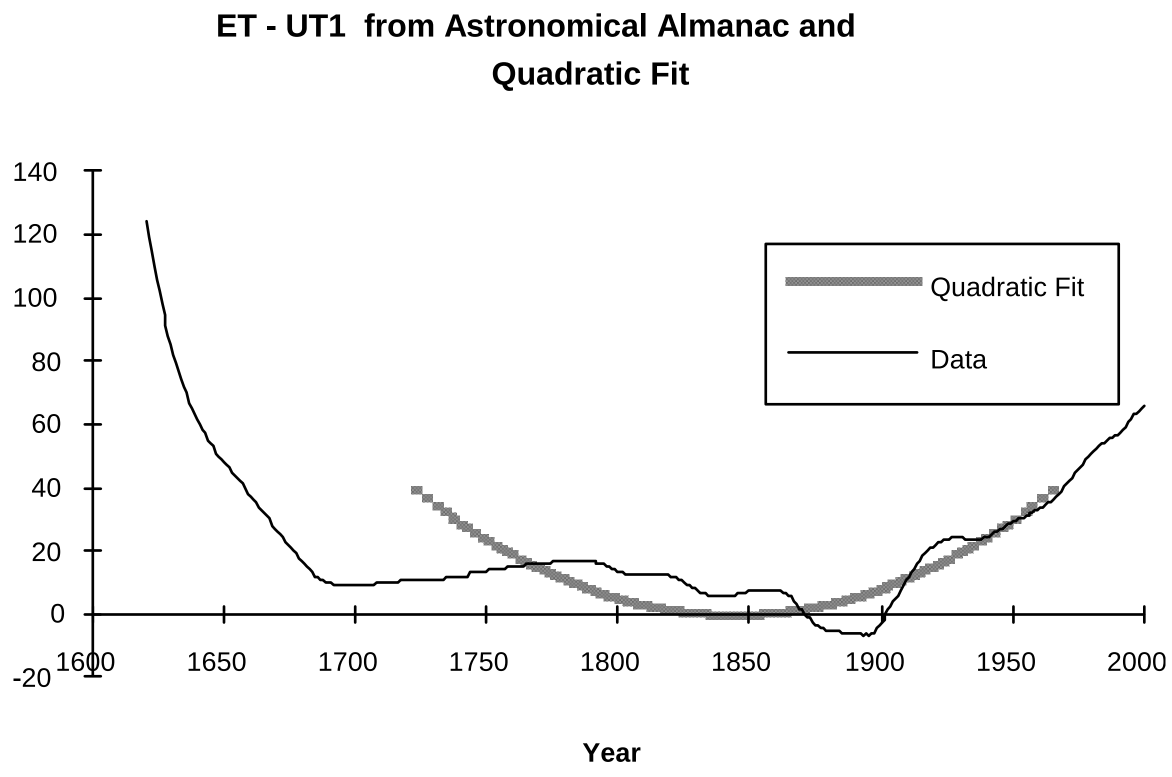


***Figure 4–2. ET–UT1 From Astronomical Almanac and Quadratic Fit***

Two different fits to the data are in the Explanatory Supplement to the Astronomical Almanac, p. 83. One is from 1650 to "the present" (probably about 1986) and the other is from 390 BC to 948 AD. The first of these is probably much better over the whole interval. The fit in the present document was produced by adjusting the fitting constants so as to fit the recent data well. Due to the large, relatively long term departures of the difference ET–UT1 from a parabola, which implies non uniform angular acceleration, it is difficult to judge what is the best fitting method.

If the fit done here is accepted as a working basis, then, by differentiating the fit for ET–UT1, the mean rate of UT1 relative to TAI, at present and in the next decade, is given to a good approximation in 1994 by

- dUT1/d(ET) = dUT1/d(TAI) = 1.0 - 0.00226 sec/day = 1.0 - 2.616 x $10^{-8}$ = 0.999999974
  - The U.S. Naval Observatory publishes, in its series 7, equations that can be used for short term predictions of UT1–UTC. By the next increment of the toolkit, this or another method based on the above procedure will be used to provide useful estimates in cases when data files have not been maintained on schedule, although, of course, warnings will be issued.

## 4.7 Accuracy Issues; Tradeoffs in Accuracy and Storage

The accuracy available with different time streams is discussed in this section.

### 4.7.1 Basic Rationale

In designing the algorithms and software to maintain necessary time information for the SDP Toolkit, several advantages and disadvantages needed to be balanced. Any time stream that is kept as a real number (floating point, double precision, integer or whatever), rather than in ASCII or some equivalent coded form, is economical in terms of storage, but is generally difficult for the user to interpret. The best case for ready interpretation is probably the Julian Date, because it is well known. Julian days are measured from noon; Greenwich mean solar time, 4713 BC. This introduces two problems, however:

a. Even with a double precision word in C or FORTRAN on most platforms (exception: The Cray with "long double" in C or "double precision in FORTRAN) the best accuracy that can be preserved is of the order of a few milliseconds. During that time the spacecraft moves many meters.

b. The use of the Julian Day to represent several differently defined time streams, such as UT1, UTC, TDT, TAI, and TDB could lead the user astray. The Astronomical Almanac carefully warns users to specify which time stream is under consideration when using Julian Days.

The penalties for using ASCII coding are obvious:

a. A large storage requirement for (say) 27 characters.

b. One cannot perform arithmetic on ASCII coded numbers.

Because of these problems, the Toolkit was designed to use one ASCII time per invocation, with an array of TAI offsets.

### 4.7.2 Accuracy Issues in Geolocation

Turning to issues in geolocation, as a touchstone, we note that machine accuracy is important in getting the spacecraft position and attitude correct, as well as the Earth orientation. An error of 1 second time in Earth rotation is equivalent to an error of up to 1/3 mile, while an error of one *milli*second for the spacecraft amounts to a positional error of 7 meters (on top of any actual ephemeris error). The most trying part, numerically, of the ECR to ECI transformation and its inverse is simply that the time is kept as a double precision number in Julian days. This is used to get Greenwich Mean Sidereal Time. Since one full rotation gives no change in the position (in terms of axial rotation) the whole number part of the Julian day, six figures, loses its significance. Only the fractional part can be counted on for accuracy in diurnal rotation. Thus a machine accuracy of 14 figures reduces to 8, and one of 15 to 9. But a day contains 86,400 seconds, so a part in $10^8$ is nearly a millisecond. It is largely because of this problem; two double precision numbers are used to carry Julian Date, enhancing the accuracy about two million fold. (The other reason is that the internal planetary ephemeris accepts this form of number for utmost accuracy, although the Toolkit user interface works with UTC and offsets.) Problems with spacecraft position are minimized by using UTC and time offsets based on seconds from Jan 1, 1993. Even by 2007, there will have been only ~ 4 x $10^8$ sec elapsed. Thus, with 14 significant figures the Toolkit can calculate times to better than 0.01 millisecond, or less than 7 cm of spacecraft motion, using only one double precision number. The errors in UT1–UTC data being of the order 0.1 millisecond, no accuracy will be lost by using TAI in seconds from Jan 1, 1993 to get UT1jd, but it is still important to express UT1jd as two doubles (which would cause an error of order 1 millisecond)! The advantage of using seconds from Jan 1, 1993 rather that carrying all the Julian days from 4713 BC is then manifest.

## 4.8 Relationships Between the Time Streams; Algorithms

Algorithms and in some cases certain data are needed to transform among the time streams. This section presents the algorithms, and Section 6 discusses the data files and sources. In the present section, the necessary constant algorithms are given.

### 4.8.1 Important Constants; Supplementary Definitions

The SI second is the duration of 9192631770 cycles of transition between the hyperfine levels of the ground state of Cesium 133. Because of relativity, the clocks used for this standard are to be at rest in Earth fixed coordinates on the geoid.

The Julian century consists of 36525 days of 86400 seconds each of a time designated Terrestrial Time (TT) (see below); that is for our purposes the same as 36525 days of 86400 seconds of TAI or TDT. (See IAU Circular #93, Aug. 30, 1994).

The epoch J2000 is defined as JD2451545.0 = Jan 1, 2000 noon (also called Jan 1.5). As late as 1993 the "noon" in question had been defined in TDB, but the IAU decided in 1994, at General

Assembly XXII, in resolution C7, that J2000 will be defined to be at the noon of Jan 1, 2000 in a time called "TT." At J2000, this time is equivalent, for our purposes, to 0 hours TDT or 0.0000994 seconds before midnight TDB. See IAU Circular 93. TT is essentially TDT or TAI + 32.184 s. The difference is a name change plus a conceptual change that TT is supposed to be defined from something called Geocentric Coordinate Time (TCG). The latter time is a theoretical construct that is constant on surfaces that are not parallel in 4–space to those of TDB, but for all practical purposes, in the near Earth environment, it is simply TAI times a scale factor intended to correct clock rates from the geoid to distances far from the Earth, but ignoring the effects of the sun. The net effect of all the definitions is to reduce TT once again to TDT, at least in the near Earth environment, while TCG runs at a rate that is uniform but not equal to that of TT. The difference, then, in changing the definition of J2000 from TDB to TT will be trivial for Toolkit applications because the only effect is less than a micro arc second motion of the J2000 axes.

- 1993 Jan 1 midnight UTC = 2448988.5003125 as a TAI Julian Date. (The 0.5 is from the noon/midnight difference in UTC and Julian Date, while the 0.0003125 is the 27 accumulated leap seconds on Jan 1, 1993.)
- TDT = TAI + 32.184 s by definition and agreement (IAU, USNO, IERS)
- GPS = TAI–19 s by definition by the USAF

### 4.8.2 Algorithms

#### 4.8.2.1 Relationship of TAI and UTC

The algorithms to transform between the various time streams are all built into the SDP Toolkit software, but most are listed here with references. In a few cases, we demonstrate how to do transformations that are not done in the toolkit, such as from TDB to TDT.

- TAI = UTC + leap seconds (after Jan 1, 1972) (see Appendix A)

#### 4.8.2.2 Relationship of UT to Other Times—Older Data

From Jan 1, 1961 to Jan 1, 1972 the Naval Observatory has supplied ramp functions (exhibited in Appendix I) to estimate what the leap second correction ought to have been to convert UT1 to TAI (there was no UTC, and there was no TAI, but we can regard TDT - 32.184 s as a surrogate for TAI. Note that TDT was then just called "ET" [ephemeris time] and was not distinguished from TDB).

The Toolkit implements the ramp functions so that older data sets, with times in UT1, can be brought into accurate correspondence with EOS data.

After Jan 1, 1972, UT1– UTC was kept and can be retrieved if necessary (see the reference to the file "*utcpole.1972to1979*" below). Prior to Jan 1, 1972 UT1 was used at a time standard, except that in some cases UT2 was used (see the 1994 Astronomical Almanac, p. L2). At present, it is assumed that Earth Observing measurements prior to 1972 were not at such resolution that the difference between UT1 and UT2 would be important. This difference can, however, be of the order of 30 milliseconds, during which spacecraft motion is appreciable. Therefore, users wishing to validate or rework the geolocation for older (pre 1972) data sets ought to determine what time signals were used for the ephemeris. The Toolkit does not supply UT2, but the algorithm (from the IERS Bulletin B) is given in Section 4.8.2.7.

#### 4.8.2.3 Relating TDT and TAI

- TDT = TAI + 32.184 s
- TAI = TDT - 32.184 s

#### 4.8.2.4 To Relate TDT and TDB to Each Other

Let g = mean anomaly of the Earth (radians) (angle of the Earth from its perihelion based on its mean motion):

- g = 6.24008 + 0.017202*(jedTDB - 2451545.0)

In the foregoing, we ostensibly need TDB to do the computation. In principle, when starting from TDT this would require recursive or iterative programming because one does not have TDB right away. The difference is always < 1.7 milliseconds. In that time interval, g does not change by more than (0.17202 radian * 0.0017 sec)/86400 sec = 3.3 x 10 $^{-9}$ radian. (The factor 1/8400 is from differentiating jedTDB, which is in days, with respect to changes measured in seconds). Clearly, this difference is of no consequence, so we can get g from TDT instead of TDB in the argument. Then

- jedTDT = jedTDB - [0.001658 * sin(g) + 0.000014 * sin(2*g)]/86400.0

  (use TDB in the equation for g)
- jedTDB = jedTDT + [0.001658 * sin(g) + 0.000014 * sin(2*g)]/86400.0

  (use TDT in place of TDB in the equation for g)

This algorithm is from p. B5 of the 1995 Astronomical Almanac, and is separately derived in Appendix I. Note that the Explanatory Supplement to the Astronomical Almanac, on p. 43, gives a much more complicated Equation (2.222–2). That equation contains many more corrections, for example, from the Lunar potential. It is somewhat misquoted from the paper: T. Moyer, Celestial Mechanics Vol. 1, p. 32 (1981) Eq. 46. The first square root bracket ought to terminate *before* the quantity "e sin(E),” but the overscore was allowed to extend erroneously over those terms. In any case, the formula given in the present document is fully accurate enough for the Toolkit. The equations relating TDT and TDB are, as has been intimated, approximate, because certain periodic terms (mainly due to effects of the moon) are omitted. Also, the assumed eccentricity of the Earth's orbit appears to be slightly out of date, but the equations are accurate within 2 microseconds. Of course, even a few milliseconds error would have little effect on EOSDIS work because TDB is used only in the Solar/Lunar/Planetary ephemeris!

To get TAI from TDB we can first get TDT from TDB, then TAI from TDT.

### 4.8.2.5 Obtaining UT1

UT1 = UTC + (UT1–UTC) where UT1–UTC is available in tabular form from the IERS or the USNO. Tables are accurate to ~ 0.0001 s (~ 0.0003 s before 1988). The Toolkit maintains the differences in the file “utcpole.dat”. The data begin at 1979–06–30, and need to be updated weekly (see below). A file of older data, from 1972–01–01 to 1979–06–29 will be supplied with Toolkit 4 as "*utcpole.1972to1979*". This file would require minor reformatting to be used. These data were supplied by the U.S. Naval Observatory, courtesy of Dr. Dennis D. McCarthy.

### 4.8.2.6 Mean and Apparent Greenwich Sidereal Times

Greenwich Mean Sidereal Time is defined by the equation (Explanatory Supplement to the Astronomical Almanac p. 50 Eq. 2.24–1, or 1995 Astronomical Almanac p. B6):

$$\text{GMST of 0 hours UT1} = 6^h{:}41^m{:}50^s.54841 + (8640184.812866\ T + 0.093104\ T^2 - 6.2 \times 10^{-6}\ T^3)\ \text{sec},$$

where T is the time in Julian centuries of 36525 days of UT1 from 2000 Jan 1, 12 h UT1. From 0 hours, UT1 increases with GMST at the rate of 1/1.00273790935 mean solar seconds per sidereal second. For more decimals, see the *IERS Standards*, p. 30 (Ed. Dennis D. McCarthy, USNO, 1992). The base quantity $6^h{:}41^m{:}50^s.54841$ can also be expressed as 24110.54841 seconds. The factor 1.00273790935 is the ratio 365.242190/366.242190 of the number of solar days in a sidereal year to the number of sidereal days in a sidereal year. The ratio of the mean solar and sidereal days is also given on p.B6 of the 1995 Astronomical Almanac as 1.002737909, the same number. Again, it is emphasized that although time units are used in deriving GMST, in the Toolkit it is always in radians.

The above ratio should not be confused with the value 1.002737811906, which is the ratio of the UT1 day to the period of sidereal Earth rotation (Explanatory supplement to the Astronomical Almanac, p. 52). This ratio is slightly less because the sidereal period of Earth rotation is slightly more than a sidereal day. The reason for this discrepancy is the precession of the equinoxes; the equinox moves slowly retrograde (West) 0.0084 seconds a day, causing the Meridian transit of the equinox to come slightly sooner than if the equinox was fixed. (In this regard, remember that while the precession of a top on a table is prograde, the precession of the equinoxes is retrograde, because the mean torques of the sun and moon cause a torque that tries to align the Earth's rotational pole with the ecliptic.) The sidereal rotation period of the Earth is needed only for dynamical purposes or in transforming velocities.

The difference between GAST and GMST is called the "equation of the equinoxes." There are at least three different versions extant for this equation; several will be discussed below. The exact form used is not crucial provided that everyone uses the same definition, because in the last analysis, the Earth orientation is measured and tabulated (see the remarks on UT1, the file "utcpole.dat" below, and Sections 6.4.1.1 & 6.4.1.2). So long as all measurements are based on a common definition, the tabulated measurements will enable anyone to recover the true Earth orientation in J2000. According to the Explanatory Supplement to the Astronomical Almanac, p. 116, it is equal to the right ascension of the mean equinox referred to the true equinox and equator of date. According to the Astronomical Almanac and the Supplement, the Greenwich apparent sidereal time is found from the mean through

GAST = GMST + Dpsi * cos(obliq_true)

where Dpsi is the nutation in longitude, and obliq_true the true obliquity of the ecliptic of date. But the author has been informed by the USNO that, following Woolard (*Astronomical Papers of the American Ephemeris and Nautical Almanac,* Vol **XV**, Part I, U.S. Government Printing Office, 1953), the IERS uses

GAST = GMST + Dpsi * cos(obliq_mean)

Thus some of the results of the USNO (which produces the *Almanac)* and the IERS may be inconsistent. The inconsistency is at less than the 0.1 arc second level and need not concern us further. In point of fact, to obtain the USNO equation, the spherical triangle made up from the true equator of date, the mean equator of date, and a normal from the mean equinox of date to the true equator of date (Figure 3.222.1 of the *Supplement*) has been approximated as a plane triangle. If this approximation is removed, the equation would become

GAST = GMST + arctan[tan(Dpsi) * cos(obliq_true)]

The error here is even smaller, not exceeding a micro arc second, owing to the smallness of Dpsi. Ignoring this problem, the IAU decided in 1994, at General Assembly XXII, in resolution C7, that after Feb. 26, 1997, the definition of GAST will be

GAST = GMST + Dpsi cos (eps0) + 0".00264 sin (Omega) + 0".000063 sin(2*Omega),

where eps0 is the mean obliquity of the ecliptic (called obliq_mean above), and Omega is the mean longitude of the ascending lunar node. The mean longitude of the ascending Lunar node is (according to the 1980 IAU theory)

Omega = 125 degrees 02 minutes 40.480 sec arc - (5 revolutions + 134 degrees 08 minutes arc 10.539 seconds arc) * t _ 7.455 $t^2$ sec arc + 0.008 $t^3$ sec arc,

where t is time in Julian centuries of 36525 days of 86400 seconds dynamical time since J2000. *Note that the number 125 is misprinted as 135 in the Supplement to the Astronomical Almanac, on p. 114, Table 3.222.2*. Reference: IERS Standards (1992) p. 32. This misprint was discovered in intercomparing algorithms and has been verified by Dr. K. Seidelmann at the USNO.

The author is in the process of ascertaining if the nutation in longitude in the foregoing equation is to be the measured one or the one from the IAU theory of 1980—to EOSDIS accuracy it will make no difference, but it’s nice to know. Again, it makes little difference what is done here so long an everybody understands and follows, because in the end, data are taken and used to define the Earth motion at this fine a level of detail. Of course, if these definitions are optimal, then the theoretical expressions will handle most of the problem and the data tables will be smooth, and the entries small.

#### 4.8.2.7 UT2

UT2 is not provided in the toolkit, but can be calculated if desired by users as follows.

The following material is excerpted from the guide to Bulletin B of the IERS. It is identical with Equation (2.57–1) on Page 85 of the Explanatory Supplement to the Astronomical Almanac:

> UT2 can be derived from UT1 by adding the following conventional annual and semi annual terms.
>
> UT2–UT1 = 0.0220sin(2*3.141593*t) - 0.0120cos(2*3.141593*t)
>
> - 0.0060sin(4*3.141593*t) + 0.0070cos(4*3.141593*t),
>
> the unit being the second and t being the date in besselian years.
>
> t = 2000.000 + (MJD - 51544.03) / 365.2422.
>
> Tables of UT2–UT1 are available from the Central Bureau of the IERS on request.

The MJD can be based on UT1 or UTC in this algorithm. It can be seen that this algorithm requires knowledge of UT1, and, as such it is useless except, when UT1 is available, to interpret data tabulated in UT2. For continuous periods in between leap seconds, however, it is possible to treat the algorithm as follows: Remembering that UT2 is an older approximation designed to smooth UT1 by removing seasonal variations, one could assume that the change in difference UT2–UT1 is an approximation to the change in the difference r*UTC–UT1, where r is the ratio of the UTC (or TAI) second to the mean solar second, r = dUT1/d(ET) ~ 1.0 - 2.616 x $10^{-8}$ = 0.999999974. This could only be applied to segments of time not including any leap second boundary, and only to changes, because the base difference (baseline for UTC) is set by the BIH. By this means, one could estimate the change in UT1–UTC from the last good datum until the next leap second, if the file "*utcpole.dat*" were out of date. This method is not particularly recommended, and is sure to be inferior to the actual methods used by the IERS and USNO for their predictions, but it is probably better than using nothing. The method is similar to one in the USNO series 7 and a comparison is under study.

## 4.9 Required Data Files; Maintenance

To transform from UTC to any of the continuous time streams (UT1 as a Julian Day, TAI, TDT, TDB, GPS) one needs the leap seconds data (*leapsec.dat*) which must be up to date. To transform between times (Sidereal, UT1) based on Earth rotation and any other form, one needs the data for UT1–UTC (*utcpole.dat*), which must be up to date.

### 4.9.1 Access to Data Tables

This section explains access to the tables and interpolation of them (where appropriate).

#### 4.9.1.1 Leap Seconds—File leapsec.dat (shown in Appendix A)

The data table for leap seconds is accessed with the function PGS_TD_LeapSec() which is called by numerous Toolkit functions. It must be updated when new leap seconds are announced. The function PGS_TD_LeapSec() (which does not have a user guide entry) implements the ramp functions shown in Appendix A for times before Jan 1, 1972. It is important to note that leap seconds are announced in advance, in Bulletin C of the IERS, which is also provided by the USNO. Therefore, an entry in this table is marked "ACTUAL" as soon as the leap second is announced, which is normally at least three months in advance. Little maintenance problem ought to occur, except that the long term predictions will get out of step with actuality, and with predictions of UT1. By the time of the release of the next increment, a remedy for this problem of deriving the leap second predictions from the UT1 predictions will be provided.

#### 4.9.1.2 UT1–UTC—File utcpole.dat

The data table, utcpole.dat, for UT1–UTC is accessed with the function PGS_CSC_UTC_UT1Pole(), also not in the user guide. This function reads tables only a small part of which can be shown here. The polar motion data are carried with the values of UT1–UTC as all three are required for Earth orientation. The values are issued each month by the IERS in Bulletin B, and weekly by the USNO in IERS Bulletin A in series 7 (sic—the IERS Bulletin issues from the USNO and not the IERS). Final values are marked "f" at the right, intermediate values "i,” and predictions "p".

The tables are interpolated linearly in the function PGS_CSC_UTC_UT1pole(), which does not have an entry in the User Guide. Extensive studies showed that the interpolation error was much less than that already quoted for the entries. The methodology was to interpolate on 2 day intervals, testing results against the intermediate day. The error is assumed to decrease a factor 4 when the interval is halved, as is appropriate for linear interpolation of smooth data.

### 4.9.2 Maintenance of Data Tables:

It is currently planned that ECS will maintain the required files.

# 5. Celestial Body Access Tools

This section discusses the tools that obtain celestial body coordinates in ECI or in SC coordinates. Here, celestial bodies mean the sun, the moon, and the eight planets other than Earth.

## 5.1 Introduction

This section explains the following tools.

Tools:

- PGS_CBP_Earth_CB_Vector()
- PGS_CBP_Sat_CB_Vector()

### 5.1.1 Organization

This section is divided as follows:

5.1—Introduction:

5.2—The J2000 Reference System–ECI

5.3—The ECI True of Date (TOD) and ECR Coordinate Systems

5.4—The Toolkit version of the JPL DE200 Ephemeris

5.5—Geometric and Corrected Planetary Coordinates

5.6—Algorithms for Parallax and Aberration

### 5.1.2 Summary and Overview

To set the stage for the Celestial Body Position (CBP) tools, this section begins with a few facts about J2000, other reference systems, precession, nutation, geometric coordinates, aberration, light travel time effects, and related matters, including orders of magnitude. In reading this section, remember that although a few EOSDIS users need positions of celestial bodies to only to low accuracy, some are quite sensitive to a celestial object's intruding into the field of view. For these users, we need to reduce errors to about one second of arc. This number is picked because although the spacecraft attitude will initially be good only to a few seconds arc, by the use of control points, etc., users may be able to reduce the error to about a second arc. For a spacecraft 700 km up, looking at a slant range of 2000 km, one second arc translates to more than 20 meters error; at nadir it is 3.5 meters. For perspective, note that the JPL ephemeris and good star catalogs are valid to a few hundredths of a second of arc, although for the outer planets the accuracy might be somewhat less over long time spans. See X.X. Newhall, E.M. Standish, and J.G. Williams, *Astron. & Ap*. **125**, 150–167 (1983), E.M. Standish, *ibid* **233**, 252–271 (1990).

The original JPL access and interpolation software was translated to C. The numerical Celestial Body identifiers were replaced with: PGSd_SUN, PGSd_MOON, PGSd_MERCURY, PGSd_VENUS, PGSd_MARS, PGSd_JUPITER, PGSd_SATURN, PGSd_URANUS, PGSd_NEPTUNE, or PGSd_PLUTO. Earth is omitted as all our software gives celestial body positions in geocentric or spacecraft coordinates.

## 5.2 The J2000 Reference System—ECI

This is an inertial coordinate system (neglecting the Earth's acceleration). It has a pole where, according to the IAU standard theory of precession, the Earth's rotational angular momentum is predicted to point at noon, TDB, Jan 1, 2000 and an equinox where the Earth's equatorial plane is predicted to intersect the ecliptic at that time. There is a very small uncertainty in what is meant, due to errors in nutation's theory, but so long as all parties use the same theory, no problem ensues. Technically, the J2000 coordinate system is centered in the barycenter of the solar system. If its axes are attached instead to the center of the Earth, one gets the "ECI" system, in which the spacecraft ephemerides for EOS will generally be provided by the Flight Operations Segment (FOS) or FDF.

Concluding this section, it essential to remember that the spacecraft ephemeris will be defined in J2000. Thus our work on Earth orientation is all designed to define the Earth's position in terms of the ECR system in J2000. Pixel location on the Earth's surface involves locating the spacecraft and the look vector in ECR; getting the sun angle at the surface involves bringing the DE200 sun vector to ECR. Stellar and planetary positions, on the other hand, are usually desired in J2000 so that they may be related to on–board instrument apertures, or other ports or sensitive surfaces. Thus, they can be left in J2000; of course, they may be brought into ECR if needed by our same transformations.

## 5.3 The ECI True of Date (TOD) and ECR Coordinate Systems

At any date and time, it is possible to define various reference systems based on the Earth's true rotation pole at that time, and an equinox based on the intersection of the Earth's equatorial plane with the ecliptic at that time. These frames are called "True of Date" or "TOD". Furthermore, it is possible to use the mean pole and equator, ignoring nutation, in place of the true pole and equator, leading to what might be called the "mean of date" system. The TRMM ephemeris will be provided in TOD coordinates, and will be placed in J2000 by the Toolkit[5]. Other than in the case of the TRMM spacecraft, the TOD systems are of limited interest to EOSDIS. They are useful for ground based observers who need to tie their observations to standard objects, or the Earth's rotation at a specific moment. One application of some interest to EOSDIS is that certain historical data sets may be based on B1900 or B1950 instead of J2000. Observers wishing to transform to or from these coordinates will need to use software for precession and nutation. The

[5]The TRMM ephemeris was handled incorrectly in Toolkit 4. The TRMM documents stated that it would be in "J2000 True of Date," which was interpreted as J2000, but really stands for True of Date, according to telephone conversations with GSFC TRMM staff. This will be corrected in Toolkit 5.

main use of TOD arises in obtaining ECR coordinates by the use of Greenwich Sidereal Time and polar motion. The TOD system is, for our purposes, a bridge system between ECI and ECR, because its instantaneous rotation pole is along the current angular velocity. ECR is discussed in Section 6.

### 5.3.1 Celestial Equator and Ecliptic

Imagine the sky as a huge sphere around the center of the Earth. A plane through the Earth's center and normal to the Earth's instantaneous rotation axis is the true celestial equator of date. If the rotation axis is set to a theoretical value at a fixed date (such as J2000), including precession but not nutation, it is the mean equator of that epoch. Thus, mean and true are always distinguished by the fact that "mean" includes precession only, with nutation removed by using theoretical expressions, while "true" means actual with no correction for nutation. As the Earth goes round the sun, the sun appears to trace out a curve against the background of stars; that curve (corrected a tad for lunar and planetary perturbations so that it is a plane curve) defines the ecliptic plane. It can be called the plane of the Earth's orbit (more strictly the orbit of the Earth–moon barycenter). It’s normal defines the ecliptic pole. The ecliptic plane changes (against the background of stars) much more slowly than the celestial equator and we will not be concerned here with such changes, which are due to perturbations of the Earth's orbit. The invariable plane is the plane through the center of mass of the solar system and perpendicular to its angular momentum; as suggested by its name, it changes much less than the ecliptic, if at all. (Tidal torques of the Galaxy and relativistic effects could slowly change the "invariable" plane; but to the best of the author's knowledge, such effects have not been measured yet.) The ecliptic is not the invariable plane, and its motion has been both measured and predicted.

### 5.3.2 Obliquity of the Ecliptic

The Earth's axis is inclined at about 23.4 degrees from the normal to the ecliptic; the angle between the celestial equator and the ecliptic is called the obliquity of the ecliptic. Short period variations in it are called nutation in obliquity. According to IAU Circular # 99, Dec. 9, 1994, the most accurate prediction of the inclination at J2000 from DE245 is 23 degrees, 26 minutes, 21.409 seconds of arc, or 23.43928 degrees. The official IAU value is 0.039 arc seconds larger. The official IAU value, 0.409092804 radians is used in the toolkit.

### 5.3.3 Precession

Due to various torques, mainly those of the sun and the moon, the tip of Earth's axis of rotation, projected on the celestial sphere, executes a circle of size about 23.44 degrees about the ecliptic pole, in a period of about 26,000 years. This does not affect J2000 tabulated positions. It is of importance to observers who need to locate an object in coordinates "of date”; i.e., coordinates tied to the pole and equinox at the time of observation. Along with nutation, it is also used for transformation between reference systems such as B1900, or B1950 and J2000. See the 1992 Explanatory Supplement to the Astronomical Almanac (U.S. Naval Observatory), pp. 99–105.

### 5.3.4 Nutation

Nutation is any variation in the Earth's rotational pole that departs from precession. An attempt has been made to divide between the two phenomena at a time scale ~ 18.6 years, the period for one retrograde motion of the line of nodes of the moon's orbit on the ecliptic. In practice, the precession algorithm is kept fixed for many years or decades, until modified by the IAU or other competent body, while various theories of nutation may be presented and adopted in time periods of a few years to a decade or so. Nutation in obliquity refers to the change in inclination. Nutation in longitude is the motion of the equinox along the mean ecliptic. In simpler terms, you can think of nutation in obliquity as a slight change in the tilt of the Earth's axis, and nutation in longitude as a slight advance or retardation in the precessional motion. Nutation can be decomposed into various terms all of shorter period than precession, and the nutation tables that are used by our  nutation program essentially just combines amplitudes and periods with sines and cosines to reconstitute a Fourier decomposition of the motions. See the *1992 Explanatory Supplement to the Astronomical Almanac* (U.S. Naval Observatory), pp. 114–115. There is an error in the last line of Table 3.222.2 on p. 114, where the number 135 degrees ought to be 125 degrees.

## 5.4 The Toolkit Version of the JPL DE200 Ephemeris

Barycentric coordinates are coordinates centered at the barycenter of the solar system, and, for our purposes, having axes aligned with those of J2000.

JPL Provides the ephemerides of the sun, moon planets and asteroids to observatories and data centers around the world. These ephemerides are based on many decades of optical and radar observation; in recent decades, one can also say that long baseline interferometric radio observations are used as well, for these provide the best measures of the orientation of the Earth. The observations were fitted with a dynamical set of equations based on Einstein's corrections to Newton's laws of motion. At the same time that the best fit orbits for the planets are determined, best fit values for the masses are also obtained (although in some cases, fly–by data may be used as preferred masses).

Because the ephemerides are derived from dynamical equations, they are expressed initially in Barycentric coordinates. EOS observers will require the coordinates of celestial bodies as seen from the spacecraft or the terrestrial lookpoint. In the latter case, only the sun and the moon are expected to be of interest. The toolkit provides the vector from Earth center to any body, but corrects only the moon for the fact that the terrestrial lookpoint is not Earth center. (In the case of the sun, the apparent direction differs only by 8.8 sec arc. The error in using the geocentric position is much less than that due to refraction, but users concerned about the error can themselves correct for it.) In the case of the moon, the correction (which can be as much as a degree) is performed in the function  PGS_CSC_ZenithAngle(). In the case of viewing from the spacecraft, the Toolkit function PGS_CBP_SCtoCelestialBody() corrects in all cases for the displacement of the spacecraft.

There are several ways coordinates could be provided, even in the Barycentric J2000 system. The coordinates could be provided either in "geometric" form, which means with the same Barycentric time used for both the target planet and the observer. This position is useful only for those doing dynamical calculations, and is not provided by the Toolkit. The useful position is the observed position, corrected for parallax, aberration, and light travel time, as discussed below.

Appendix G discusses the original specifications of the ephemeris.

## 5.5 Geometric and Corrected Planetary Coordinates

The original ephemeris is based on J2000 coordinates, and is "geometric". That means that the positions and velocities of the planets and sun are provided at the same instant of Barycentric Dynamical Time (TDB) throughout the solar system.

### 5.5.1 Geometric Coordinates for the Planets:

The point of this section is to distinguish "geometric" coordinates, which cannot be observed, from what would actually be observed. Geometric coordinates are defined in J2000 in regards to constant Barycentric Dynamical Time (TDB), and certain rectangular and Right Ascension (RA), declination (DEC) systems defined by the IAU. Apart from some very small general relativistic corrections, which would, to our accuracy, affect only objects nearly in line with the sun, we may think of the space coordinates as rectangular cartesian coordinates with the Z axis towards the J2000 pole, the X axis towards the J2000 equinox, and the Y axis so that (X,Y,Z) make a right handed orthonormal triad. Other choices besides "geometric" are coordinates that are corrected for aberration and/or light travel time. The usual JPL ephemerides do not supply these, for several reasons. Practically speaking, the “DE200” ephemeris (originally from JPL) is intended not only for observers on the Earth or in the near Earth vicinity, but for spacecraft anywhere in the solar system. Aberration and light travel time would be different for different spacecraft. Furthermore, the results are all produced by powerful computer programs that integrate dynamical equations based on geometric positions, so it is most direct and safest to publish results in these coordinates.

### 5.5.2 Relativity Corrections

There are additional small corrections from geometric coordinates to observed coordinates due to general relativity. For objects observed more than a degree from the solar limb, the correction is < 1 sec arc. It is omitted in the current increment of the toolkit. Whether future increments will include this correction in a later release is To Be Determined (TBD); it would seem that users are unlikely to be concerned about a 1 second arc error for some star or planet when they are looking only a degree from the solar limb. There are also tiny (< 1 arc second) gravitational effects for objects nearly in line with Jupiter, Saturn, etc., but we are not providing the positions of the satellites of these planets, and it is virtually impossible that some chance alignment with a planet will cause a measurable apparent displacement of a background object.

### 5.5.3 Rings and Satellites

A more serious concern to users would be to allow for rings and satellites, which could intrude in the FOV. Our planetary models used for PGS_CBP_body_inFOV() utilize radii from pp. F2–F3 of the 1995 Astronomical Almanac. This problem is discussed in detail in Section 8.6.

### 5.5.4 Aberration

Aberration of stars is due to the Earth's motion around the barycenter of the solar system. It is a function of location of the star on the sky and of the time and date; it approximately reverses every six months. (For a circular orbit, and no perturbations, it would exactly reverse.) The mean place of a star is defined so that aberration is removed, and it varies only due to proper motion. (Annual parallax is also removed.) The apparent place includes aberration and parallax, and varies during the year. Planets and the sun suffer similar aberration. The catalogs of stars, planets, etc., (sun, moon) are all based on TDB and J2000. Therefore corrections to be made must be made with these positions and this reference frame as the starting place. Aberration and light travel time effects can be treated separately. For example, aberration would be considered as due only to the motion of the observer through the J2000 reference system. By way of example, suppose we were looking at Venus. If we worked all the analysis in an inertial reference frame moving with the Earth, the aberration would be zero, and all the correction would be obtained by correcting Venus' geocentric position for light travel time. This would be an approximation, because instead of aberrating the vector from Venus "then" to Earth "now" we construct a reference system moving with the Earth, approximate it as inertial, and throw all the corrections into the light travel time from Venus.

### 5.5.5 Aberration—Practical Considerations

A decision had to be made either to calculate aberration for the planets and sun, based on the Earth's velocity in J2000 beforehand, and imbed it in a corrected ephemeris, or to calculate it on the fly. The problem of doing it in advance is that it is relative to the Earth center. Actually, one needs the position of a celestial body as seen from the Earth's surface (look point) or from the spacecraft. The additional aberration due to the Earth's surface velocity is under 1" arc; but that due to spacecraft velocity is nearly 5" arc. For this reason, and to facilitate checking the ephemeris against standard comparison data, it was decided to do the corrections at execution time.

### 5.5.6 Parallax

The spacecraft is not at Earth center. Therefore, the apparent position for a celestial body is different from what it would be if viewed from Earth center. The tool Spacecraft_to_Celestial_Body corrects for this effect, which ranges from ~ one degree for the moon to a second of arc or less for the outer planets. (When Jupiter is at opposition; the displacement is about 2 seconds of arc.)

## 5.6 Algorithms for Parallax and Aberration

An observer on the Earth or in an orbiting spacecraft will perceive a position for the planet that differs from the "geometric" ephemeris position for three reasons—parallax, aberration, and light travel time, which we discuss in turn.

### 5.6.1 Parallax

The observer's position is not at the center of the Earth. The necessary correction is called the parallax correction. The equation is:

**x_corr = x_geom - (x_obs - x_Earth)** (5.6–1)

where

**x_corr** is the geometric position corrected for parallax, **x_geom** the raw position from the Ephemeris, **x_obs** the observer's coordinates, and **x_Earth** the coordinates of the Earth's center.

### 5.6.2 Aberration

The observer's velocity in the Barycentric frame is not zero; it is nearly equal to the orbital speed of the Earth, about 29.9 km/s. This velocity causes aberration of the observed light so that the target planet seems slightly closer to the tip of the observer's velocity vector. The correction for this effect is called the aberration correction, or the "stellar aberration" correction, because it applies to stars and galaxies as well as planets. It has to do only with the observer's velocity relative to the barycenter, and has nothing to do with the motion of the target (source of the light). The reason for this is purely definition. As we know from relativity, only relative motions can be defined well, but, in the present case, the "true" direction of the light rays is defined in the Barycentric reference frame.

The equation for this correction (*Supplement to the Astronomical Almanac*, p. 129) is

$$\mathbf{p1} = (\mathbf{p}+\mathbf{V}/c)/|(\mathbf{p}+\mathbf{V}/c)| \qquad (5.6\text{–}2)$$

where **p1** is the corrected unit position vector**, p** that before correction, and **V** the Barycentric velocity of the observer. The result is accurate to first order in V/c, which is adequate for toolkit requirements.

We shall include only the velocity of the Earth's center and, if the observation is from a spacecraft, that of the spacecraft; we'll ignore the Earth's surface velocity. The correction for aberration to the spacecraft reference frame is done by having PGS_CBP_Sat_CB_Vector() pass the ECI vector from Earth to the celestial body to PGS_CSC_ECItoSC(). The latter function corrects for both parallax and aberration.

### 5.6.3 Light Travel Time

The motion of the target planet during the time that it took for light to travel from it to the detector. The correction for this effect is called the correction for light travel time.

If the Earth's motion were rectilinear, then effects of aberration and light travel time could be combined, by using the total velocity difference between source and receiver, and considering light travel time only. This amounts to putting the planet's orbit in the reference frame of the Earth. In that reference frame, there is no aberration for an Earth bound observer. The planet's position relative to the Earth must be calculated in the past, however, i.e., with the relative motion back–dated to the light emission time. This approximation is called the "planetary aberration" approximation, as it cannot be used for stars (whose Barycentric velocities are poorly known, and whose geometric position is, anyway, of little interest).

Various equations for this correction are in the Supplement to the Astronomical Almanac (A.A.), pp. 133–134. Let **uB**(t0) be the planet's Barycentric position at time t0, when the light left the planet, and let **eB**(t), **veB**(t) be the Barycentric position and velocity of the Earth at the observation time t. Let **vB**(t) be the planet's velocity at t; to the required accuracy we can ignore the change in velocity from t0 to t. Also, tau is the light travel time from planet to Earth, t - t0. The solution involves iteration, because the light travel time tau depends on the distance, which is not known accurately until the position is corrected. Therefore the Equation (3.255–4) of the Supplement

$$\mathbf{P1} = \mathbf{uB}(t0) - \mathbf{eB}(t) - tau*( \mathbf{vB}(t) - \mathbf{veB}(t)) \qquad (5.6\text{–}3)$$

must be solved by iteration. On the first trial, tau is estimated as | **uB**(t) - **eB**(t)|/c, where **uB**(t) **eB**(t) are evaluated at the observation time. On later trials, the value of t0 is obtained from

$$t0 = t - tau$$

where

$$tau = | \mathbf{uB}(t) - \mathbf{eB}(t)|/c. \qquad (5.6\text{–}4)$$

The SDP Toolkit uses the approximation in Equations (5.6–3, 5.6–4) (assumption of constant relative velocity). According to Dr. George Kaplan of the U.S. Naval Observatory, it is good within better than 0.1 second arc. The accuracy of the approximation is due to the smallness of the accelerations of the planets. By experimentation, it has been found that to get answers to within that accuracy, only one iteration is needed. Thus, only one is taken.

# 6. Coordinate System Conversion (CSC) Tools

## 6.1 Introduction

This section explains the relationships among the Toolkit coordinate systems and the following tools.

Tools:

- PGS_CSC_ECItoECR()
- PGS_CSC_ECRtoECI()
- PGS_CSC_ECRtoGEO()
- PGS_CSC_GEOtoECR()
- PGS_CSC_ECItoSC()
- PGS_CSC_SCtoECI()
- PGS_CSC_ORBtoSC()
- PGS_CSC_SCtoORB()
- PGS_CSC_ZenithAzimuth()

### 6.1.1 Organization

This section is divided as follows:

6.1—Introduction

6.2—Primer on Earth Motion and Coordinate Systems

6.3—The Aberration of Light; light travel time

6.4—Coordinate Transformations–Algorithms

### 6.1.2 Summary and Overview

The Toolkit provides four rectangular coordinate systems—ECI, ECR, Spacecraft and Orbital. The Toolkit provides a family of geographic based coordinate systems, defined by user–selected or defined Earth axes. These tools are derived from a wide variety of user requirements. Thus the only suggested usage is described in Section 1.3.

This section first discusses the basis, and then the algorithms and software.

## 6.2 Primer on Earth Motion and Coordinate Systems

This section describes the reference frames and coordinate systems used in the EOSDIS SDP Toolkit. It pertains to all the Celestial Body Position (CBP), Ephemeris Data Access (EPH), and Coordinate System Conversion (CSC) tools. Also see Section 5.1 for more details on ECI. The ECI system is more fully discussed in Section 5.

### 6.2.1 The ECI Coordinate System

ECI Reference Frame: In the Toolkit, the ECI system means the J2000 celestial reference frame. It's North pole or Z axis is along the predicted rotation vector of the Earth at midnight, Jan 1, 2000 AD (JD 2451545.0); its X axis is toward the vernal equinox on that date, and it's Y axis comprises a right handed orthonormal triad with the X and Z axes, in the order X,Y,Z. This frame is nearly inertial; its origin has a small acceleration (~ 0.5 cm/s$^2$) because the Earth goes around the sun, but its axes remain aligned with an inertially fixed set of directions, as well as can be established. It will be called "inertial". This reference frame is the standard for the newer and current JPL planetary ephemerides, such as DE200, which is used in the Toolkit, and it coincides with the FK5 (Fundamental Catalog 5) reference frame used by recent star catalogs (W. Fricke, *Astron. and Astrophys.* **107,** L13, 1982). The AM and PM spacecraft ephemerides will be supplied to ESDIS in J2000 coordinates.

### 6.2.2 The ECR Coordinate System

The ECR reference frame has its North Pole or Z axis at the Earth's geographic North pole (not necessarily exactly the true Earth rotation axis), its X axis along the meridian of Greenwich England (semi–circle of zero longitude), and it's Y axis forming an orthonormal, right handed triad. It is related to the ECI True of Date (TOD) frame by three rotations: x and y displacements tabulated in seconds of arc in the EOSDIS table "utcpole.dat" (See the 1994 Astronomical Almanac, p. B60), and an axial rotation equal to Greenwich Apparent Sidereal time plus a correction due to UT1–UTC, which will now be described. The order of these rotations has been established by the International Earth Rotation Service, and, proceeding from the celestial TOD frame to the terrestrial, it is (z,y,x), the z rotation being about the rotation axis and the x and y being displacements of the crust from that axis. (See p. B60 as above). In reversing the transformation, the order must be reversed. The signs of the angles also reverse in that case. The sign conventions are on p. B60 and are mentioned in the code (ECI to ECR and ECR to ECI); they will not be repeated here. EOSDIS functions obtained from Dr. E. Myles Standish at JPL provide Greenwich Mean Sidereal Time. This locates the Greenwich meridian relative to the *mean* equinox of date. This must be corrected to the apparent equinox of date as described below, but in addition, the axial rotation of the Earth's crust is not quite constant, due to motions in the core, ocean and atmospheric currents, etc. The Earth's precise angle of rotation about its axis is found from tables of UT1–UTC, where UT1 is a historical descendant of traditional methods of timekeeping. These methods measured the angle of the sun from the true equinox of date (including nutation), not the mean equinox. (See Equation 2.242–6 on p. 53 of the Explanatory Supplement to the Astronomical Almanac, and related text nearby.) Thus, it is necessary to correct Greenwich Mean Sidereal Time to Greenwich Apparent Sidereal Time by applying the

Equation of the Equinoxes; implicitly we are indeed using the true equator and equinox of date. The UT1–UTC correction, which is tabulated in seconds of time in the EOSDIS table "utcpole.dat" is applied by using UT1 in the function "sidt2.c" adapted from Standish. The other approved method would be to correct the rotation by the fraction of a full circle represented by 1.002737909 * (UT1–UTC) seconds, where one full rotation is 86400 seconds. The factor 1.002737909 is the conversion from solar to sidereal time rate. (See the 1992 Explanatory Supplement to the Astronomical Almanac (U.S. Naval Observatory), Figure 2.25, p. 55 for details.) Finally, the polar motion corrections are applied. The orders of magnitude of these corrections are as follows:

***Table 6–1. Corrections to Earth Orientation***

| **Correction** | **Typical Value In Equivalent Meters (Worst Case)** | **Error (Equiv. Meters)** |
|---|---|---|
| Equation of the Equinoxes | 500 | 0.3 |
| UT1–UTC | 500 | see below |
| Polar Motion | 15 | see below |

## 6.2.3 Relationship of ECR and ECI Systems

The ECR and ECI coordinate systems have the same origin, at the center of the Earth. They differ in only in their orientation and state of rotation. The poles of the two will be aligned almost exactly at J2000, but at other epochs they differ. What is more important, ECI is an inertial system, for our purposes, while ECR rotates with the Earth.

The very slight difference in the rotation pole and the Z axis at J2000 is due to departures of nutation from the IAU 1980 nutation theory used to define J2000, and is expected to be negligible (see the remarks on EOP data, below). Polar motion has no effect on the relationship. It may seem odd that a particular nutation theory is used to define an “absolute” reference frame. The reason is, to the best of the author's understanding, that the reference frame has been established so that the nutation averages to zero when a large segment of time around J2000 epoch is used in the average. Thus, to get any datum at any other time into J2000, some theory must be used, and that means that a nutation theory is implicit. At the instant J2000, the mean axes agree with the J2000 reference axes. The basic ideas are that J2000 is a system of type "mean"; to go between any other mean axes and J2000 one precesses only. To go between true of date and J2000 one nutates as well.

The functions that transform between ECR and ECI take into account the following:

a. Precession of the Earth's axis (due to torques of the sun and moon)—The motion is such that the North rotation pole makes a circle of angular radius 23.4 degrees about the ecliptic pole in about 26,000 years.

b. Nutation of the Earth's axis (a shorter—term variation, of smaller magnitude than precession in the long term, and due mainly to the same sources)—The 1980 International Astronomical Union (IAU) approximation is used. The time scales vary from a few days to a few months. The small departures from the IAU theory, called "EOP" motions and measured by the International Earth Rotation Service (IERS) are not incorporated; the consequent error is only of order < 1 m, and varies slowly, on the time scale of many months. To incorporate these data would, in any case, result in inconsistency with GSFC/FDF ephemerides for the spacecraft, because GSFC does not use the EOP corrections.

c. The existence of leap seconds and the values, kept in the file "leapsec.dat" enter these calculations in a minor way—The leap seconds are important in transforming from UTC to dynamical time, which is then used to access the functions for precession and nutation. If the leap seconds file is missing or out of date, PGS software will use an approximation. As a result, only small errors in precession and nutation, of order < 5 meters, will ensue. The leap seconds file ought to be kept up to date for best accuracy; a warning message will issue if it is out of date and an error message if it is missing. The leap seconds are also used in the CBP tool group, and there if the file is absent or out of date substantial errors will ensue, especially for the apparent positions of the moon and sun.

d. Diurnal Earth rotation—This consists of a steady part and small corrections that must be obtained in tabular form from the IERS or USNO (U.S. Naval Observatory). The steady rotation, 0.000072921151467 radians/(SI second) is built into the SDP Toolkit software. The corrections are kept in the file "*utcpole.dat*" that must be maintained at each Distributed Active Archive Center (DAAC). The corrections are comprised of secular and variable parts. The secular part contains any error due to the use of the fixed numerical constant given above for the rotation speed. The variations in the corrections are due to motions in the Earth's oceans, atmosphere, and core. Although the random part of the rotation amounts to only about 10–30 meters, the tables of UT1–UTC and polar motion that must be used to correct this random variation also deal with the fact that the TAI/UTC clock rate is not compatible with mean Earth rotation, resulting in the slow accumulation of leap seconds. The data for UT1–UTC embody an underlying sawtooth function that compensates  for this problem. Therefore, in the absence of current data, the error can be as much as 450 m near the equator, less at the poles. For this reason, maintenance of the tables of UT1–UTC is important for those interested in critical geolocation. (We are working on a "workaround" that may alleviate this problem.) The reader who is interested in the problem should refer to the document on time streams, especially to the schematic figure comparing UTC, UT1, and TAI. The hand drawn curves of UT1 and UTC can only suggest the sawtooth shape of the difference function. UTC–UT1 slowly climbs a ramp, punctuated by random wiggles, until a leap second; then it drops below 0, and re–commences its climb.

e. Polar motion—This is a small motion (of order 0.2” - 0.4” arc or 6 to 12 meters) of the Earth's crust in relation to its rotation as a solid body. The motion (formerly called "variation in latitude" but now called “polar motion”) is expressed in terms of motions of the geographic pole as displaced from the rotation pole. (In other words, a pylon at the geographic North pole would be observed to point slightly off the instantaneous rotation pole.) The cause is, again, motions of the oceans, atmosphere and core. The correction is expressed as an "x rotation" and a "y rotation" as defined on p. B60 of the 1994

Astronomical Almanac. Any portion of the crustal motion that cannot be expressed in terms of polar motion is, of course, expressible as a change in angle of rotation about the pole, and therefore is expressed in terms of UT1 (item 3). The polar motion data are carried in the same file as UT1–UTC, namely *utcpole.dat*. The error due to unavailability of the polar motion data is only of order ten or so meters, however, which is far less serious than missing UT1 data.

#### 6.2.3.1 Errors in Transformations Between ECI and ECR

There is a small uncertainty in the Equation of the Equinoxes equivalent to up to 30 cm due to the fact that the IERS (following the method of Woolard in 1953) uses the mean obliquity to define nutation in longitude, while the U.S. Naval Observatory uses the true obliquity. See Section 4.8.2.6

The UT1–UTC values can be true measured ones or estimated, as described in Section 4.

Errors in polar motion and UT1–UTC are shown in the polar motion table in "*utcpole.dat*". The errors are assessed in Tables 1.2 and 1.3.

### 6.2.4 Other Earth Centered Rectangular Systems:

It is quite possible to define inertial systems analogous to our ECI but based on epochs other than J2000. Common systems previously in use include B1900 and B1950. ("B" is for "Besselian" as opposed to "Julian"). These coordinate systems have their Z axes coincident with the Earth rotation poles in 1900 and 1950 AD, respectively, and their X axes along the vernal equinox of those epochs. The transformations between these systems and J2000 can be performed by using standard methods for precession and nutation. The older coordinate systems are of interest to those who analyze data based on those epochs; mainly ground based optical observations of stars or planets. For this reason, the lower level PGS Toolkit functions that deal with precession and nutation have not, at this release, been included in the User Guide.

### 6.2.5 Geodetic Coordinates

Tools:

PGS_CSC_GEOtoECR( )

PGS_CSC_ECRtoGEO

Geodetic coordinates describe the same physical system as ECR, in a variant form that resembles spherical polar coordinates, but with three major differences:

a. Latitude is used in place of "colatitude"—the angle from North. (Longitude is identical with the azimuthal angle of spherical polar coordinates)

b. Geodetic latitude is used in place of geocentric—While geocentric latitude is the angle between the Earth's equatorial plane and the vector from Earth center to the point under consideration, geodetic latitude is found differently. It is determined by dropping a normal from the point of interest to the flattened spheroid that represents the Earth, and measuring the angle between that normal vector and the equatorial plane. Near the pole and equator, geodetic and geocentric latitude are nearly equal. At mid–latitudes they differ by up to 10 minutes of arc. Algorithms for transforming between the two are in numerous texts.

c. The altitude h off the Earth spheroid is used in place of the distance to Earth center, as the radial variable.

The transformation between rectangular and geodetic coordinates involves the size and shape of the Earth. The geometry is shown in Figure 6–1. The Earth's equatorial radius is denoted A, and the polar radius C. The geocentric latitude is denoted $\phi$ and the geodetic $\phi$'. The distance of a point from the Earth's axis is sometimes denoted r. Thus the horizontal line in the figure can be thought of as the r axis in cylindrical coordinates, $(r,\lambda,Z)$, where $\lambda$ is the longitude.

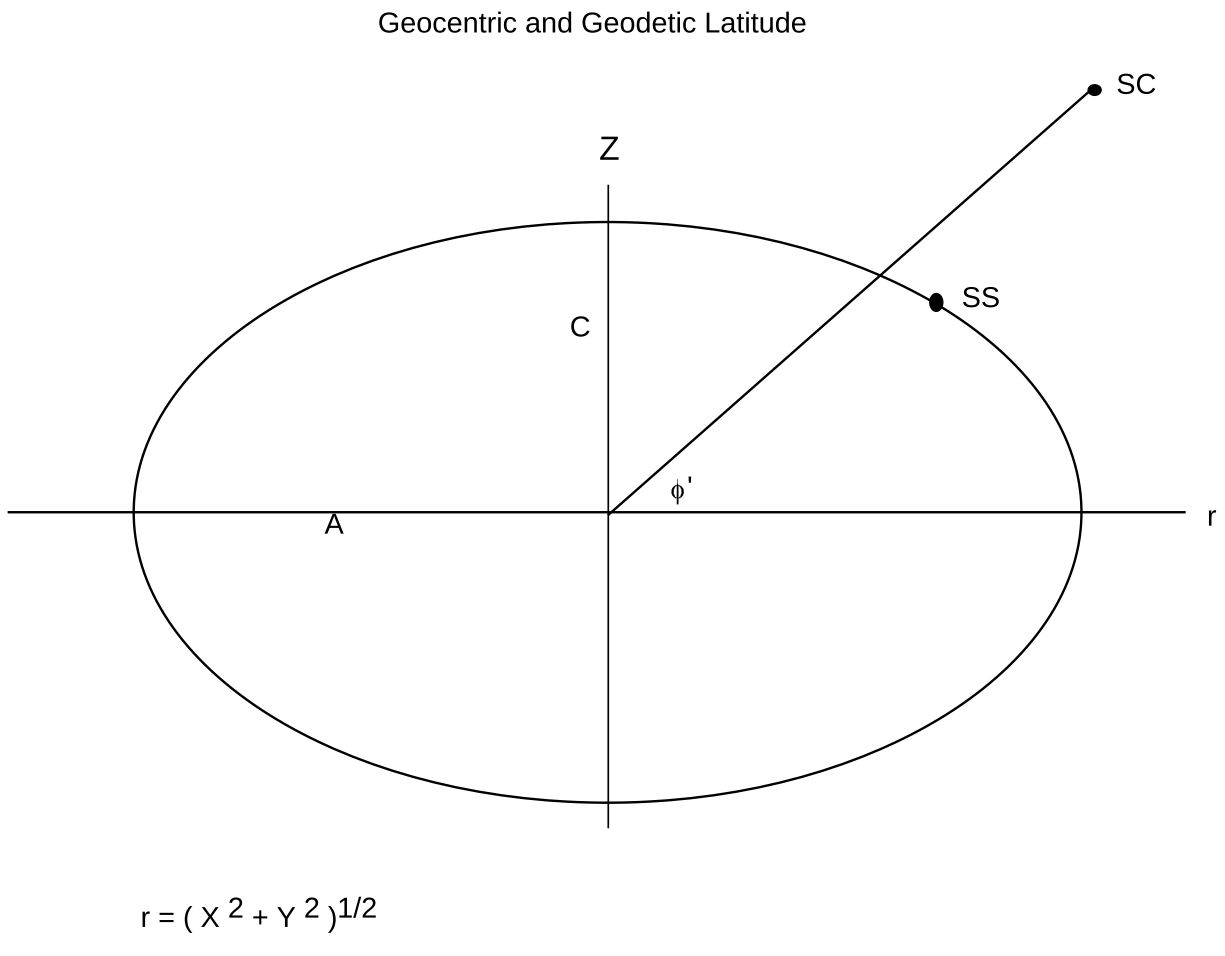


***Figure 6–1. Geodetic and Geocentric Latitude of a Spacecraft (SC) at Altitude h***

### 6.2.5.1 Figure of Earth and the Geoid

For Toolkit purposes the Earth is modeled as an ellipsoid. An ellipsoid can be triaxial or an ellipsoid of revolution, normally called a spheroid. In the SDP Toolkit the figure of Earth will be described as a spheroid, but called the "ellipsoid" in deference to common usage. Its equatorial radius will be denoted "A" or, in the software, "equatRad_A". Its polar axis will be denoted "C" or, in the software, "polarRad_C". (This notation leaves "B" for use as another equatorial axis later if it is decided to use triaxial figures.) The polar radius is less than the equatorial by about one part in 298. Numerous Earth models are discussed in Table 4.242.1 of the 1992 *Explanatory Supplement to the Astronomical Almanac* (U.S. Naval Observatory), p. 220. Additional data can be found in the *WGS84 Handbook*, p. 7–12, and the *IERS Standards* (D. McCarthy, USNO) p. 19. Toolkit 4 contains three models in the file "*earthfigure.dat*"; the WGS 84, which is the Toolkit default, the MERIT model, and a model called "new_intl" that was included somewhat arbitrarily because it is in the Toolkit ancillary tool data base (SDP Toolkit Users Guide for the ECS Project). This model apparently originated in a JPL data set called GCTCP and is rather large. Its original provenance is unknown. The WGS 84 equatorial radius is 6378137 m, and the flattening factor (A–C)/A is 1/298.257223563. The relationships between axes, flattening factor, and eccentricity are spelled out in more detail in Section 6.4.3.1.

The user can edit the file "*earthfigure.dat*" to change, delete, or insert models. The "new_intl" model will be replaced in Toolkit 5 by GRS–80, which has the same equatorial radius as WGS 84, but a flattening factor of 1/298.257222101, which is recommended in the *Explanatory Supplement* for accurate work in terrestrial transformations. Users most concerned with ultra–precise geolocation should refer to the *IERS Standards* (Dennis D. McCarthy, Editor. USNO, 1992). Tables therein show the transformations among different global and local data sets, which also includes a displacement of the origin as large as 0.517 meters in some cases—an effect the Toolkit cannot model. Station coordinates of many radio observatories used to determine Earth rotation, with coefficients for tidal displacements are in the IERS Standards, pp. 70–109, but the altitudes do not seem to be included. The latitudes and longitudes are given to the nearest 0.0001 degree, or about 11 m linear measure North–South, 11 m * cos($\phi$') East–West. Coordinates of many astronomical observatories (optical and radio), including the altitudes, are in the *1995 Astronomical Almanac* (U.S. Naval Observatory), pp. J6–J 15, but are given only to the nearest 0.1 minute of arc (equivalent to 190 m accuracy North South, 190 m * cos($\phi$') East–West).

The geoid refers to an equipotential of gravity (including centrifugal force); its normal defines the way a local plumb bob hangs. The geoid may depart from the ellipsoid by tens of meters. There is a map of the geoid in *Spacecraft Attitude Determination and Control*, by J.R. Wertz (Reidel, Holland 1985 p. 125). The map shows heights of the geoid above the ellipsoid as stated in the caption, but the associated text on the following page confusingly refers to a “variation” of “77m *above the geoid* near New Guinea.” What is meant is an elevation of the geoid *above the ellipsoid.* The geoid also falls a little more than 105 m below the ellipsoid in the Indian Ocean, and these two extremes appear to bracket the variation. The GEM–8 model is somewhat out of date. A current model, JGM2 on a 70 x 70 grid and, when ready, the forthcoming 360 x 360 gridded model can be obtained from Space Geodesy Branch of NASA Goddard Space Flight Center, code 926. Tables for recent models are in the *IERS Standards*, Chapter 6 and the 1992 *Explanatory Supplement to the Astronomical Almanac* (U.S. Naval Observatory), pp. 227–232.

The Toolkit CSC functions do not include geoid data, but users may wish to check the Ancillary Data Access Tools (SDP Toolkit Users Guide for the ECS Project).

### 6.2.6 Topocentric System

Tool:

PGS_CSC_ZenithAzimuth( )

At any Earth point there is defined a local horizontal plane, a zenith vector, and vectors in the horizontal plane that point North, East, South and West. This is the topocentric system. In it, one speaks of a zenith angle (or its complement, the elevation off the horizontal plane), and the azimuth, which is measured Eastward from North.

The function PGS_CSC_ZenithAzimuth() calculates the zenith angle and azimuth of any ECR vector. It corrects the moon vector for geocentric parallax and has available an approximate refraction correction, including not only zenith angle, but physical displacement of the intersection of the ray with Earth.

SDP Toolkit software has been checked against USNO tables for correctly determining the sun Zenith angles so as to predict sunrise and sunset, and by personal observation of several amateur observers to test the sun azimuth at sunrise and sunset in various locations.

### 6.2.7 Barycentric System:

The ephemeris underlying the PGS_CBP....() tools is in solar system Barycentric (center–of–mass) coordinates, but the SDP Toolkit software converts the origin to Earth center or SC as required. The lower level functionality is available by examining the functions themselves, but is not in the user guide.

### 6.2.8 Spacecraft Coordinates:

Each spacecraft has its own coordinate system defined by fiducial markings. The AM series and TRMM have similar setups; later spacecraft are not yet defined. There is a diagram for the AM spacecraft in Figure 1–2 of the *Earth Observing System (EOS) AM–1 Flight Dynamics Support System (FDSS) Requirements Specification* (553–FDD–94/018ROUD0, published by the Mission Operations and Data Systems Directorate, GSFC, June 1994). For AM and TRMM, in nominal rest attitude (for example, all orbital–to–spacecraft Euler angles being zero) the spacecraft coordinate system is aligned with the orbital system. The x axis is opposite to the thruster, so that it is essentially along the velocity, other than small variations in the velocity due to orbital eccentricity, maneuvers, or excursions within attitude control limits. The z axis is toward nominal nadir—the side of the spacecraft on which the instruments are mounted. The y axis is the cross product of z and x. The individual instrument platform reference directions may depart from nominal due to biases. The instrument locations and mounting angles, gimbal angles of moving parts, etc., are assumed to be known by users. Problems of flexure and aging are not handled by the Toolkit. The Toolkit will keep track of any changes in attitude through the quaternions.

The roll axis is x, the pitch y, and yaw z. The senses of roll, pitch and yaw have been assigned arbitrarily in Toolkit 4 in accord with the usual definitions (Section 3.3). It is planned by Toolkit 5 to make further sign checks against the platform interface documents as these develop.

It should be noted that in some cases, roll, pitch and yaw may be re–referenced when the spacecraft is flying out of nominal attitude—preliminary indications are that TRMM will do this. The Toolkit will compensate for flying mode information in the attitude quaternions, but will report Euler angles as received and interpolated.

## 6.3 The Aberration of Light; Light Travel Time

As discovered by James Bradley in 1745, when two coordinate systems are in relative motion, a light ray will not appear to have the same direction in both systems unless it is along or opposite to the relative motion. The angle of aberration is roughly proportional to the sine of the angle between the ray and the velocity, and its sense is such that each of the two observers senses the light as coming more nearly from in front of her relative velocity than is true in the other system. The maximum size of the angle of aberration, in radians, is approximately v/c, where v is the relative velocity and c that of light.

The aberration due to the Earth's velocity is about 20 seconds of arc, so that at six month intervals a typical star will appear to move by 40 seconds of arc. The aberration for a Low Earth Orbiting (LEO) spacecraft, relative to Earth, is about 5 seconds of arc. The effect in Earth observing is that points near Earth limb (horizon) and in front of the spacecraft will appear more nearly in front, and those near the Earth limb but behind will appear less behind, or more nearly underneath. Obviously, in the backward direction, it is in principle possible to miss the Earth limb and look into space by accident, if one does not allow for aberration; in practice, effects of geoid, topography and atmosphere will generally dominate. But the effect is systematic and can result in distortions of the geography up to 40 meters, so it is accounted for in certain tools.

The following functions have corrections for aberration:

- PGS_CBP_Earth_CB_Vector()
- PGS_CBP_Sat_CB_Vector()
- PGS_CSC_GetFOV_Pixel()
- PGS_CSC_SCtoECI()
- PGS_CSC_ECItoSC()

The ECRtoECR and ECRtoECI functions do not concern themselves with aberration, basically because the two coordinate systems have the same origin. (Rigorously, if the transformation between the two systems was performed very far from Earth center, there is an effect, but light rays do not travel on precisely straight lines in the rotating ECR coordinate system.) (C. Moller, *The Theory of Relativity,* Cambridge University Press, 1952, pp. 240–245.) Any aberration effect between the two systems is, however, very small, under 1 second arc, in the near Earth environment, meaning out to about 3 Earth radii.

The methodology for PGS_CBP_Earth_CB_Vector() and PGS_CBP_Sat_CB_Vector() is discussed in the section on light travel time and in the relevant ATBD. The correction is done for PGS_CSC_GetFOV_Pixel() to improve the accuracy in the lookpoint determination. In the last two cases it is done for consistency with PGS_CSC_GetFOV_Pixel(), and on the basis that EOSDIS is in the business of remote sensing, so that most information will travel by light rays. Therefore the apparent position of an object in ECI as seen from the spacecraft is more important than its instantaneous geometric position. There is virtually no way to determine the geometric positional relations in real time, and therefore little sense in providing them.

The last two functions PGS_CSC_SCtoECI() and PGS_CSC_ECItoSC() do not make the aberration correction for points within 120 m of spacecraft center, on the supposition that the user may wish simply to get the alignment of some part of the spacecraft in ECI, or conversely. The user is cautioned in that case that the results are purely geometrical! In other words, any user–supplied pixel look vector is corrected for spacecraft motion in PGS_CSC_GetFOV_Pixel(), but if PGS_CSC_SCtoECI were applied to a diaphragm, baffle, or reticle on the spacecraft it would not correct the alignment for aberration. Thus the geometric relationships of different parts of the spacecraft will be preserved. This is a convenience if the user maps a vector such as the sun vector from ECI to SC, because such a vector is faithfully mapped with correction for aberration of the ray, and thus its alignment with a feature on the SC will be accurate. The only problem that could arise would be if the user attempted to use a physical vector (not a unit vector) connecting two points in the SC reference frame to define a look vector direction, and then to map that to ECI (and perhaps later to ECR). The transformation would be purely geometrical, with no aberration correction, and would be inconsistent with the routines that use aberration, as well as with the laws of nature.

The function PGS_CSC_SubSatPoint() is a purely geometric routine, with no correction for aberration. Aberration was omitted for several reasons, such as simplicity, the smallness of the correction at nadir, the simplicity of the requirement, and for consistency with FDF ground track. Furthermore, it is in principle possible to use the function to define when the spacecraft passes overhead from the standpoint of a terrestrial observer.

### 6.3.1 Aberration as it Relates to the Use of Control Points

The foregoing methodology is perforce a compromise. The user interested in the highest accuracy results needs to take aberration into account. In the case of a fixed field of view and nearly fixed altitude and attitude, the use of control points would also remove the effects of aberration through bias determination. If the spacecraft is "flown backwards," as is sometimes done, then the effect of aberration on an instrument looking directly forward or aft of the roll axis reverses, and the control point data will be inconsistent, resulting in a sudden change of bias determination by up to 10 seconds arc total in LEO. If the same Earth point is viewed near nadir and at a large nadir angle, even with the same velocity, the error can be up to 4.8 arc seconds.

### 6.3.2 Light Travel Time:

Correction for motions during the time of light travel is made in three functions:

a. PGS_CSC_GetFOV_Pixel() corrects for Earth rotation during the time of flight of light from Earth to instrument when the high accuracy flag is set to PGS_TRUE. The correction is less than 3.5 m in LEO. The reason for the correction is, of course, that the ECI to ECR transformation is done at the instant of data taking, and not when the light left the Earth. The correction simply refers the transformation to the back–dated time if light departure.

b. PGS_CBP_Earth_CB_Vector() corrects for light travel time along with aberration by back–dating the time. These corrections are needed because the ephemeris (originally from JPL) is "geometric," i.e., based on positions that at are common instants in TDB, or Barycentric Dynamical Time.

c. PGS_CBP_Sat_CB_Vector() corrects for the additional aberration due to spacecraft motion; it does this by invoking PGS_CSC_ECItoSC(), which adds the aberration due to spacecraft motion to that due to Earth motion. This linearized method of combination is obviously not fully consistent with special relativity, but any error is of order not exceeding $v1*v2/c^2$, or a 0.0005 arc seconds where v1 is the Barycentric Earth velocity and v2 that of the spacecraft relative to Earth, while the total correction can easily be 25 arc seconds.

## 6.4 Coordinate Transformations—Algorithms

### 6.4.1 Transformation of Position and Velocity Between ECI and ECR

Tools:

- PGS_CSC_ECItoECR()
- PGS_CSC_ECRtoECI()

These tools perform a coordinate system rotation. The functions PGS_CSC_ECRtoECI() and PGS_CSC_ECItoECR() each transforms both position and velocity. Position is transformed in the obvious way, by rotating the vector. In transforming velocity, one needs to take account of both instantaneous angle of rotation, which causes the axes of the two systems not to be aligned, and the velocity difference induced by the Earth's rotation. For example, a point on the Earth's surface that is quite stationary in ECR is moving in ECI with the Earth's surface speed, while a spacecraft that is Northbound in ECI has an East to West motion superposed on the northward motion in ECR, because the Earth turns under it. To get ground track velocity and Doppler signals correct it is necessary to transform the velocity, then, so as to account not only for the different directions of the coordinate axes, but also for the apparent change in velocity induced by the rotation. This is done in the Toolkit functions for objects closer than 500,000,000 m; beyond that the velocity is set to zero on output. By this means, the functions will work correctly as far away as the moon, but the Toolkit avoids returning an absurd velocity for the sun or another star. The velocities of very distant objects cannot be defined reasonably in a reference frame that rotates with the Earth; because in such a frame they go around the Earth once a day.

### 6.4.1.1 The Transformation of Velocity

The ECI to ECR transformation and its inverse are unusual among space systems software in transforming not only position but velocity. The correction to the velocity is, of course, not large, because a typical LEO spacecraft goes about 7 km/s, while the contribution of Earth rotation to the velocity is only of order 0.5 km/s. (The Earth's surface speed at the equator is about 465 m/s, but at altitude the equivalent contribution to the spacecraft velocity can be of order 500 m/s.) The equation for the transformation of velocity is

$$v' = v + \Omega \times r \qquad (6.4\text{–}1)$$

where **x** is the cross product symbol and **Ω** is the relative rotational velocity. (S. Goldstein, *Classical Mechanics*, Addison–Wesley 1950, p. 133.)

The velocity transformation must be used judiciously. It is a kinematic effect, giving the apparent velocity in the rotating reference frame. In the case of a spacecraft, the change in velocity is small and the equation is not only reasonable, but necessary. In the case of a celestial object, the equation makes no sense. The sun does not encircle the Earth once a day, nor do the other stars. The velocity comes out absurdly large. For objects more than $5.0 \times 10^8$ m (500,000 km) from Earth center, one can pass in to the ECI to ECR transformation a six–vector with zeros for the last three components, and one should ignore the velocity output, because the function does not compute the velocity in that case (the values of the velocity components may not be set on output). The cutoff distance 500,000 km is somewhat beyond the moon. Because the function is set up to transform both position and velocity in most cases, a six component vector or array of six–component vectors must be passed into and received from the functions. A later toolkit release may provide an option for using only 3–vectors for users not interested in velocity.

Because of the Earth's rotation, these transformations require the time as an input. For accurate results, the leap seconds and polar motion/UT1 data files must be present. The correction UT1 is associated with negligible changes of the angular velocity **Ω**. (A very small effect of polar motion on the local gravity has, however, been measured; it is an effect of the variation of centrifugal forces due to varying distance from the terrestrial point to the rotation axis).

### 6.4.1.2 ECI to ECR—The Nine Rotations

The ECI to ECR transformation takes a 6 vector, comprised of 3 Cartesian coordinates in J2000 and 3 Cartesian vector velocity components in J2000 into the ECR reference frame. The transformation is comprised of 9 rotations, which will be listed here. The nature of these rotations has been discussed above, and the detailed equations are given by reference to the 1995 Astronomical Almanac. The rotations break naturally into two groups: The first group, six rotations comprising recession and nutation, connect between J2000 and ECI “TOD” (True of Date); the second set is three rotations that connect between TOD and ECR.

When precession and nutation are considered separately, the 9 rotations referred to are in three groups of three. In principle, the three rotations within a group could be combined into one matrix operation or one rotation about a resultant axis by a resultant angle. The use of the existing procedure of keeping the 9 rotations distinct is based on several factors. The software was obtained from Dr. E. Myles Standish of JPL, who developed the DE200 ephemeris; it was checked by Toolkit developers, as it is not certified software. Some of the checks are comparison of coefficients against those in the *Astronomical Almanac or the Astronomical Almanac Supplement*, which is how the erroneous expression therein for the longitude of the moon was detected—see Sections 4.2.8.6 and 5.3.4. Precession is calculated with the function PGS_CSC_PRECES2() (which does not have a User Guide entry). The Toolkit function PGS_CSC_WAHR2() (which does not have a User Guide entry) calculates nutation and its rate from a "Wahr" model as approved by the IAU in 1980. This formulation contains terms of many periods, from a few days to several months. It is accurate within about 0.04" arc. Experimental Very Long Baseline Interforemetry (VLBI) and other data have validated the Wahr model and measured further small deviations called "EOP" deviations, which are tabulated by the IERS. These corrections amount to about ~ < 1 m, which might be considered material by some instrument teams. They vary quite slowly—the timescale is months. The terms are omitted in Toolkit software because:

a. They are only about 1/10 of the more rapidly varying "polar motion" terms.

b. No one else has, to our knowledge, ever included them, so our results would disagree with those from other software, and would be incompatible with FDF data to the extent that ground station ranging is used. The terms are noted at this point for possible future inclusion; the data would have to be read from the IERS or USNO servers periodically, and added to the values from the nutation model.

Additional checks were against tables in the Astronomical Almanac and against other software. Precession was checked against Section 3.211 of the 1992 Supplement to the Astronomical Almanac. The nutation coefficients were compared with those in Section 3.222 of the 1992 Supplement to the Astronomical Almanac. Nutation values and the sidereal time were compared with the 1994 Astronomical Almanac.

These checks could not have been performed directly on software that would result if the 3 rotations for precession, or for nutation, were combined. Furthermore, the entire procedure can be compared directly with equations in the 1995 Astronomical Almanac, p. B60. Next, the same component rotations can be used, in the opposite order and with the signs of the angles reversed, in the ECI to ECR and the ECR to ECI transformations. This avoids duplication of software and guarantees reversibility of the transformations. Finally, the savings in time by combining the three rotations into one matrix operation would be small. References on the motions are to the 1992 Supplement to the Astronomical Almanac (ES) or to the 1995 Astronomical Almanac (AA). The rotations, then, are:

a. precession (3 rotations) (ES, Equation 3.21–9)

b. nutation (3 rotations). (ES 3.222–3, 3.222–6)

c. axial rotation based on UT1, converted to GAST (1 rotation) (AA p. B60)

d. polar motion (2 rotations) (AA p. B60, R1 by angle -y, R2 by angle -x)

The combination of items (c) and (d) takes vector “p3” into vector “p4” as defined in the AA. The angles x and y are exactly those in the file “utcpole.dat,” (interpolated as needed, and converted from arc seconds to radians).

If the "EOP" data were used, they would be used to modify the nutation (step b) The nutation is actually applied to a vector by the function PGS_CSC_NUTAT2().

#### 6.4.1.3 ECR to ECI

This is the exact inverse of ECItoECR; the order is:

a. polar motion (2 rotations), R2 by angle x, R1 by angle y.

b. axial rotation based on UT1, converted to GAST (1 rotation)

c. nutation (3 rotations)

d. precession (3 rotations)

Not only is the order interchanged, but the sign of each angle is reversed. On p. B60 of the 1995 Astronomical Almanac, the first three rotations (items a and b above) transform from “p4” to “p3”. The two functions, ECRtoECI and ECItoECR were tested to be reverses of each other within machine accuracy (about 15 significant figures).

### 6.4.2 ECI to Spacecraft and Spacecraft to ECI

Tools:

- PGS_CSC_SCtoECI()
- PGS_CSC_ECItoSC()

These tools transform between the Spacecraft Reference Frame and ECI

#### 6.4.2.1 Concept

The transformation involves three elements: (1) rotation (2) aberration and (3) in some cases, translation. Traditionally, one thinks of this transformation as being expressed by an attitude matrix or attitude quaternions, which express the rotation between spacecraft axes and ECI. In most cases, indeed, the user wishes to transform only a direction between ECI and SC, for example, the unit sun vector, or the unit vector representing an instrument look direction. In that case, only a unit vector should be used, representing the direction, and the whole transformation is simply a rotation. We shall see that this simplification occurs only for very distant objects, such as the sun or beyond. In this Toolkit, for consistency with other algorithms and with the viewpoint that space observing, rather than just geometry, is the issue, we also correct for aberration in most cases (see the final subsection of this section).

There may be cases, however, in which a translation is involved as well. To understand this, consider the following diagram. In the diagram, the circle at the left is the Earth, the spacecraft is at the right, denoted "SC," and the X at the top is some target of interest, such as the moon, another spacecraft, or a Tracking and Data Relay Satellite System (TDRSS) satellite. The distance from SC to Earth is greatly exaggerated. The vector displacement of the spacecraft from Earth center is denoted "R"; it can be obtained in ECI coordinates from the spacecraft ephemeris tool PGS_EPH_EphemAttit(). The dashed line at the left is the ECI vector locating X, while the bold vector at the right is the SC vector locating X. The tool is designed to handle the case that the user may have ECI coordinates for item "X" and wish SC coordinates, including the actual distance and not just the direction, or vice versa (she may have SC coordinates and wish ECI values). Furthermore, even if only the directions are of interest, and not the distances, for this kind of geometry one has to take displacement "R" into account. The direction to X seems different as viewed from Earth and from the SC is due not only to the fact that the SC coordinate axes are oriented differently, but also to the displacement of the SC from Earth center. In deriving an algorithm for this case, a key item is that the SC ephemeris is known in ECI, but the Earth "ephemeris" is not known in SC coordinates; at least the PGS toolkit does not create this set of data, though it could. Accordingly, the displacement transformation must always be done in ECI. That means that for ECItoSC, one does it first; but for SCtoECI, one does it second.

In the case of a very distant object, such as a star, it would make no sense to correct for R. It is the direction that counts. Accuracy would be lost in trying to incorporate the displacement for this case. To accommodate both needs (directional and including displacement) it was decided that the displacement would be included if and only if the input vector is not a unit vector. Thus an internal logical flag is set to translate or not according to whether the input is a non–unit vector or a unit vector.

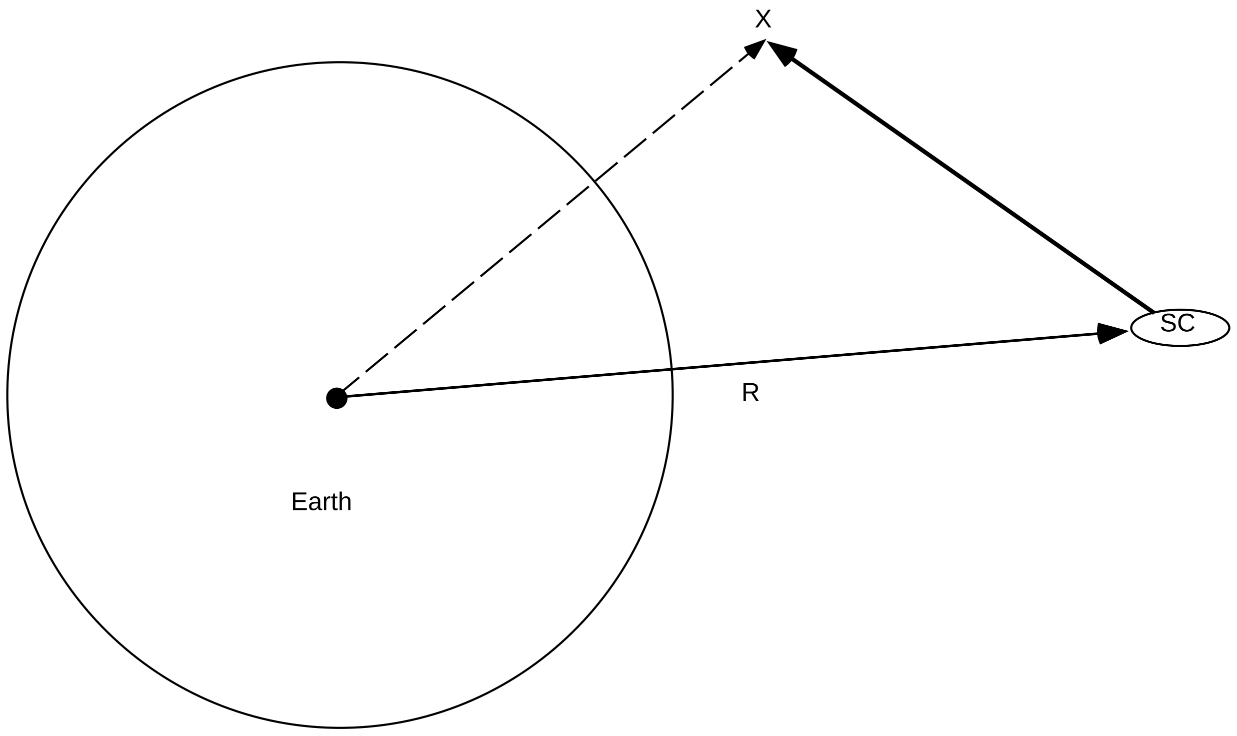


***Figure 6–2. Transformation***

Because the spacecraft is traveling at ~ 7 km/s, there is also significant aberration (~ 4.6 arc sec) between the spacecraft and ECI. Therefore it has been decided to include this correction. An explanation is given at the end of this section.

The following two subsections sketch the operations of the two functions:

#### 6.4.2.2 Operations for ECItoSC

XOUT will stand for the output 3 vector.

a. Check input tag—must be valid spacecraft

b. Check input vector X to be transformed. If its norm is between 0.99999 and 1.00001 then an internal logical flag trans is set to PGS_FALSE, otherwise to PGS_TRUE. This flag implements the translation R in Figure 6–2.

c. Check ephemeris available; get attitude (SC to ECI) quaternions. If trans= PGS_TRUE get position, R, as well.

d. If trans = PGS_TRUE, set vector XOUT = X–R where R is the ephemeris position in ECI. Otherwise XOUT = X. (these are 3 component vector replacements).

e. Apply the rotation quaternions from ECI -> Spacecraft to X (assume the attitude is stored as Spacecraft -> ECI quaternions). Use PGS_CSC_quatRotate(). First the quaternions are converted to their inverse in the sense of a reversed transformation.

#### 6.4.2.3 Operations for SC to ECI

Let XOUT stand for the output 3 vector.

a. Check input tag—must be valid spacecraft

b. Check input vector X to be transformed. If its norm is between 0.99999 and 1.00001 then an internal logical flag trans is set to PGS_FALSE, otherwise to PGS_TRUE. This flag implements the translation R in Figure 6–2.

c. Check ephemeris available; get attitude quaternions. If trans=PGS_TRUE, get position, R, as well.

d. Apply the rotation quaternions from Spacecraft -> to X (assume the attitude is stored as Spacecraft -> ECI quaternions). Use PGS_CSC_quatRotate(). Store result in a temporary vector called XTEMP.

e. If trans = PGS_TRUE., set vector XOUT = XTEMP + R, where R is the ephemeris position vector in ECI. Otherwise set XOUT = XTEMP. (these are 3 component vector replacements.)

### 6.4.2.4 Correction for Aberration—SC to ECI and ECI to SC

If the SC and an Earth bound observer look at the same celestial object (planet, star, .....) they will see it in different locations, just as a Barycentric observer will see a different location yet. The JPL Ephemeris as read by the PGS access tool corrects Barycentric locations to geocentric. The SCtoCB tool must then correct from geocentric to SC reference frame.

The correction for aberration also applies when the SC looks at the Earth itself. (There is then a small additional correction due to Earth rotation during the light travel time, up to about 3 m at large slant ranges, which we will ignore.)

Therefore, the function for transforming ECI vectors to SC coordinates contains, in most cases, the same correction for aberration as when looking at celestial bodies (those far from Earth and SC), and the function from SC to ECI contains the reverse. In particular, any ECI position is aberrated as follows in going to SC coordinates:

a. find the unit vector in the same direction

b. add **v**/c to the unit vector and re–normalize, where **v** is the spacecraft ECI velocity

c. re–scale to the original length

This must be done before any rotation but if a translation is called for that is done first.

In going from SC to ECI, the steps are as follows:

a. rotate the vector to ECI axes and normalize

b. subtract **v**/c from the unit vector and re–normalize, where **v** is the spacecraft ECI velocity

c. re–scale to the original length

Any translation is done at the end, because it is in ECI.

A problem arises, however, if we suppose that the user wishes to examine the position of the spacecraft itself, in SC coordinates. Being at the origin of SC coordinates; the SC itself ought to have coordinates (0,0,0) in its own rest system. If we take the SC ephemeris position and apply the transformation from ECI to SC to it, using the correction for aberration, the SC will be found not to be located at the center of its own coordinate system. The problem is that we have found the apparent direction of a light ray between SC and Earth center in ECI, but as measured in the SC system. We think that it is unlikely that anyone is interested in this vector. We therefore cut off the aberration effect for points deemed to be on the SC. For consistency, we also need to cut off the aberration effect when we transform the other way (SC to ECI). The cutoff is at 120 m from the nominal spacecraft center; points closer are assumed to be attached to the SC and points farther are assumed to be fixed in ECI.

The latter cutoff has a further unpleasant consequence. It means that if we use the SC to ECI transformation on the tip of the boresight of a hypothetical instrument (imagine a reticle, collimator, or gunsight in use) and on a celestial body in the center of that boresight, the celestial body position is aberrated but the boresight is not. Thus when the body is in the FOV this methodology would say it is not so. We consider it unlikely that any user would use this methodology and so encounter a contradiction of this kind! We consider the following scenarios more likely:

a. The user uses a unit vector in the SC frame to define the boresight. (In that case the aberration correction is ON, which is correct for viewing celestial bodies or the Earth)

b. The user wishes to know the geometrical location of a part of the SC in relation to the center, for example to assess the effects of gravity gradient, magnetic fields, impinging particles, etc. Although the aberration correction is small and would probably cause no difficulty in this case, what is really wanted is the geometrical position, not the trajectory of a light ray.

Incidentally, the Sub Satellite Point tool works purely geometrically, for consistency with FDF methodologies and because the requirements appeared to refer to the geometrical point. The point *observed* to be at nadir in SC coordinates will be a few meters *aft* of the geometrical one.

It should be clear from all this discussion that when and when not to correct for aberration is a complex issue, depending very much on the use to which the results will be put. Perhaps one could say that  nature is not kind to us in this regard. To cap things off, it is noted in closing that when the aberration correction is included in a transformation, it can no longer be reduced to a rotation of coordinates, even in the case that the displacement of the origin is ignored (as for unit vectors) or negligible (as in observing a distant star). The aberration correction is largest perpendicular to the velocity, and zero fore and aft, so it distorts the visual field rather than rotating it, although in a small patch of solid angle it can be approximately reduced to a rotation.

### 6.4.3 Transformation Between Geodetic and ECR

This section reviews some general features of the relationship between the Geodetic and ECR systems, and then presents in turn the algorithms that transform from Geodetic to ECR and from ECR to Geodetic.

#### 6.4.3.1 General Information

Notation:

A = equatorial radius of the Earth in meters

C = polar radius of the Earth in meters

f = flattening factor = (A–C)/A

ecc = Earth ellipsoid eccentricity

ecc2 = square of Earth ellipsoid eccentricity

h = height in meters off the ellipsoid (can be positive or negative)

hn = normalized height in meters off the ellipsoid = h/A

X,Y,Z = ECR rectangular coordinates

$\phi$ = geodetic latitude in radians

$\lambda$ = longitude in radians

r = distance from Earth geographic North–South axis, in meters

$\rho$ = r, normalized by dividing by A, the Earth equatorial radius: $\rho$ = r/A

zn = Z, normalized by dividing by A, the Earth equatorial radius zn = Z/A

NC = radius of curvature of the ellipsoid in an East–West plane

* is used for multiplication when the context is not clear from parentheses

**(*)** is used for vector dot products

atan2(Y,X) is defined as the angle whose sine is Y and whose cosine is X

ECR coordinates mean rectangular X,Y, and Z as described in the section on coordinate transformations, while Geodetic coordinates mean geodetic latitude, longitude, and altitude. *In this and other Toolkit documents and software, "latitude" will always refer to geodetic latitude unless otherwise stated.* The (r,Z, $\lambda$) system is a cylindrical coordinate system while ($\rho$,zn, $\lambda$) is a normalized cylindrical system where the Earth equatorial radius is 1.0.

The transformation from geodetic to ECR is analytic. It is described in *Methods of Orbit Determination*, by P.R. Escobal (Wiley, NY 1965) p. 29, and in the *Explanatory Supplement to the Astronomical Almanac*, p. 206. The derivation is straightforward and will not be repeated here; only the results will be presented. Refer to Figure 6–1 to see the transformation. The original point is called SC (as the case of the spacecraft is especially important). $\phi'$ is the geocentric and $\phi$ the geodetic latitude. The reason that the transformation is so simple is as follows: The point on the ellipsoid that is closest to SC (subsatellite point), namely the point SS, is easy to find in rectangular coordinates from the geodetic latitude and longitude. Also, the tangent of the geodetic latitude is the slope of the normal to the ellipsoid at SS. Therefore, the change in Z coordinate from SS to SC is just h sin( $\phi$ ), and the change in r is just h cos( $\phi$ ), where h is the altitude. The changes in X and Y from SS to SC are obviously h sin($\phi$) cos($\lambda$) and h sin ($\phi$) sin($\lambda$). The inverse transformation (ECR to Geodetic) is much more difficult because it is hard to find h and $\phi$ from r and Z.

Define the eccentricity of the ellipsoid as

ecc = sqrt(2 * f - f * f) (6.4–1)

where f is the flattening factor and “sqrt” the square root function. Or, more conveniently, let us set

ecc2 = 2 * f-f *f, (6.4–2)

which is the square of the eccentricity. (One can use the fact that (1-ecc2) = (1- f) $^2$ to simplify some expressions, but that will not be done here.)

To present the equations compactly we need a certain quantity NC that is often called the "radius of curvature in the prime vertical" (*MODIS–LEVEL–1 Geolocation, Characterization and Calibration Algorithm Theoretical Basis Document, Version 1*, 1993, p. 60; Department of *Defense World Geodetic System Handbook*, 1984, p. 7–9) and which is called the "radius of curvature in the meridian" in the *1992 Explanatory Supplement to the Astronomical Almanac*, p. 206. Neither of these designations is illuminating, latter designation being incorrect. The correct expression for the curvature in the meridian, is derived in Appendix H. To understand the difference between the different curvatures, envision the Earth ellipsoid cut by various vertical planes, all containing the local ellipsoid normal. Each slice defines a plane curve with some radius of curvature. The North South curve is in the Meridian plane; orthogonal to it is the East–West vertical plane called the "prime vertical" by some sources.

If we define

$\xi$ = 1.0/sqrt(1 - ecc2 * sin( $\phi$ )*sin( $\phi$ )) (6.4–3)

then the quantity in question is

radius_of_curvature_in_EW_plane = NC = A* $\xi$ (6.4–4)

It is easy to show that the radius of curvature of the surface of the ellipsoid in a North/South (meridional) plane is

radius_of_curvature_in_meridian = A* (1 - ecc2)* $(\xi)^{3/2}$ , (6.4–5)

which is not equal to NC. This is demonstrated in Appendix H. It can also be shown that NC is the curvature of the ellipsoid in a plane orthogonal to the meridian (i.e., an East–West, vertical plane), but that will not be demonstrated here. [It is trivial to do so using the derivative of the normal with respect to longitude and the fact that a circle of constant latitude has radius NC cos( $\phi$ ).] We can conclude that the "prime vertical" originally referred to the East–West plane. Using the quantity NC and the notation of the *1992 Explanatory Supplement to the Astronomical Almanac*, p. 206, otherwise, we simply quote the result.

### 6.4.3.2 Geodetic to ECR

$X = (NC+h) \cos(\phi) \cos(\lambda)$ (6.4–6)

$Y = (NC+h) \cos(\phi) \sin(\lambda)$

$Z = (NC*(1-ecc2)+h) \sin(\phi)$

Of course, the all–important normal vector is found from the derivative with respect to h:

**normal** = $(\cos(\phi) \cos(\lambda), \cos(\phi) \sin(\lambda), \sin(\phi))$. (6.4–7)

Note that using geodetic latitude puts the normal vector (zenith vector) in the same form it would have on a sphere, a considerable advantage.

### 6.4.3.3 ECR to Geodetic

In going from ECR to geodetic, first note that

$$\lambda = \tan^{-1}(Y/X)$$

This enables us to work in cylindrical coordinates (r,Z). Numerous algorithms exist for the transformation from ECR to geodetic coordinates. See, for example, the *1995 Astronomical Almanac*, p. K12, or the *1992 Explanatory Supplement to the Astronomical Almanac*, pp. 206–207. Most iterative schemes are slow at mid–latitudes, where the difference in geodetic and geocentric latitudes can be up to 10' arc. Their convergence can deteriorate at large geocentric distances, just where the calculation ought to be easy. The rigorous analytic transformation uses the solution to the quartic equation. The resulting complex algebra can be reduced to real operations, but the algorithm is singular at the poles; thus a small region would have to be cut out at the poles where the iterative method or another approximation would be used. The following iterative method is very simply motivated and was found to converge very quickly (3 or 4 iterations) to 11 or more significant figures—less than 0.01 mm. It does not appear to be in the literature.

The algorithm is best understood by reference to Figure 6–2. In this diagram, B is the point to be transformed, and O is the origin of coordinates (at the center of the Earth). The diagram represents a section through the Earth's axis and the object B. The algorithm is based on iterations converging on the distance w from the Earth's center to the intersection with the Earth's equatorial plane of the prolongation of the normal from the ellipsoid to the object "B" at X,Y,Z.

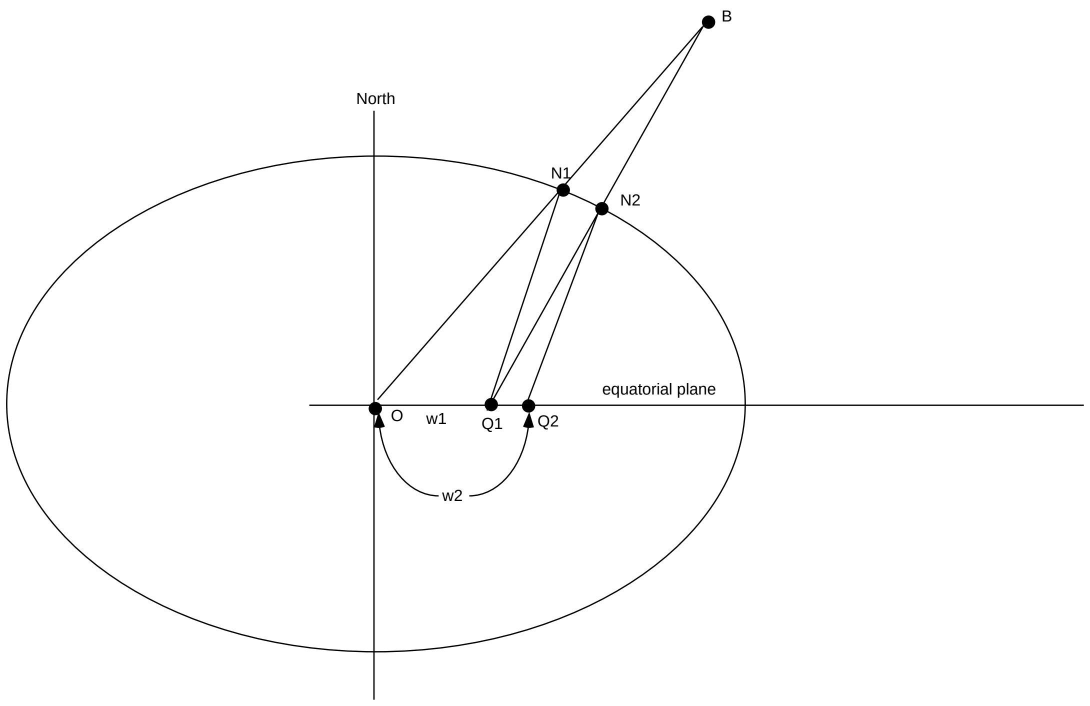


***Figure 6–3. Geometry for a Rapidly Convergent Algorithm for Geodetic Coordinates***

Note that the "while" loop is short and simple and that is has no trigonometric functions in it. It might be speeded slightly by working with the square of w instead, for then no square root would have to be taken until afterwards. If the loop is assumed to converge, so that wst = w, a quartic equation results that leads to the cumbersome analytic solution!

As usual, the longitude is found from atan2(Y,X) and there is no further separate use of X or Y; the distance to the Earth's axis is used instead. Here it is used in normalized form "ρ". Then iterations are used to get the latitude and altitude.

The algorithm begins by drawing the line BO from the point X,Y,Z to the center of the Earth at O, which defines an intersection of that line with the ellipsoid, at N1. The normal is then dropped from N1 to the Earth's equatorial plane, touching it at point Q1, and a distance w1 is computed from Q1 to Earth center, O. Then the line BQ1 is drawn from B to Q1, intersecting the ellipsoid at a new point N2, and a new normal dropped from N2, meeting the equatorial plane at Q2. Continuing this process defines a sequence of points that is found to converge rapidly to a point Q, such that the line QB is normal to the ellipsoid at N, the limit point of N1, N2..... The slope of the line QB then defines the latitude via that arc tangent function, after which the rest of the calculation is trivial. In the case of a satellite at B, the limit point N is the same as the subsatellite point SS, and B is the same as SC in the previous figure. The limiting line BNQ will make an angle with the equatorial plane equal to the geodetic latitude.

Note that f = (A–C)/A, where A and C are the equatorial and polar radii.

The problem is worked in coordinates normalized by A where:

A= equatorial Earth Radius. (6.4–8)

Thus, let

hn = normalized altitude = h/A (6.4–9)

$\rho$ = sqrt(X*X + Y*Y)/A (6.4–10)

zn = Z/A

The algorithm is as follows:

a. Test if point is inside or outside ellipsoid—in "C" notation; define a factor we will call "$\sigma$" to be 1 if outside, -1 if inside. ($\sigma$ will give the sign of the altitude.) Mathematically, let

   $\alpha$ = ($\rho$*$\rho$ + zn*zn/(1.0-ecc2)), (6.4–11)

   an expression that is positive for exterior points and negative for interior ones. The only need for this is to set the sign of the altitude, which is

b. Then define:

   $\sigma$ = 1.0 if $\alpha > 0$ or = 0 (6.4–12)

   = -1.0 if $\alpha < 0$

c. Next, initialize w, the distance of Q from O (Q stands generically for any Qi, i = 1,2,3,...), and the stored (lagged) value, wst; the latter is initialized large to force the loop to go at least one iteration.

   w = 0.0

   wst = 1.0

d. The iterations are commenced and continued until the difference between w and wst is < $10^{-11}$; a limit of 10 iterations was also set in case of non–convergence and a diagnostic is to be issued in that case, but it never happens.

   iterate while wst - w > 1.0e-11 and (niter < 10)) the following actions:

   1. initialize or reinitialize the "lagged" "stored" value, wst

      wst = w

   2. increment niter

      niter -> niter +1

   3. define a new value of w

      w = ecc2*($\rho$ - wst)/sqrt(($\rho$ - wst)*($\rho$-wst) + zn*zn * (1.0-ecc2)) (6.4–13) this is the end of the while loop!

Next we find the latitude, its cosine and sine, and the omnipresent $\xi$ (which is just NC/A). The latitude is obviously found from the slope of the line NB:

$\phi = \tan^{-1}(zn,\rho\text{-}w)$

We also need the cosine and sine of the latitude, and "$\xi$" soon in order to find the altitude:

cos_lat = cos(phi)

sin_lat = sin(phi)

$\xi$ = 1.0/sqrt( 1.0 - ecc2 *sin_lat * sin_lat ) (6.4–14)

To obtain h it is now necessary to define the location of point N on geoid at foot of normal from B to ellipsoid; i.e., the limit point of the Ni. The point N has normalized cylindrical coordinates:

$\rho$ = cos_lat * $\xi$ (6.4–15)

zn =(1.0 - e2) * sin_lat * $\xi$ (6.4–16)

By the Pythagorean Theorem, then,

h = $\sigma$ * A * sqrt((x-$\rho$)*(x-$\rho$) + (z-zn)*(z-zn)) (6.4–17)

This completes the derivation.

### 6.4.4 Transformation to Topocentric Coordinates: Zenith and Azimuth Tool

The topocentric coordinate system is set up at any terrestrial point by using the normal to the ellipsoid as a “vertical” axis, and in the horizontal plane setting up axes pointing North and East, forming, all told, an orthogonal system. The direction of any vector, such as that to the sun, or from the spacecraft, can be specified by its zenith angle and azimuth. The latter is the angle from North, measured East, of the vertical plane containing the vector to the plane of North and zenith. For example, East is at $\pi/2$ radians azimuth, and West at $3\pi/2$. It may often be desired to know the zenith and azimuth of the sun, moon, or look vector.

Tools:

PGS_CSC_ZenithAzimuth()

Function:

PGS_CSC_SpaceRefract() (to be a Tool in Toolkit 5)

### 6.4.4.1 Purpose

The purpose of these tools is to get the Zenith angle $\zeta$ and azimuth $\psi$ of the sun, moon or other celestial body, or of the look vector, at the look point or any terrestrial point (which will be called simply the "Earth point,”) corrected for refraction if desired. The displacement of the look point due to refraction can also be obtained, although additional equations are needed to convert to changes in latitude and longitude (Section 6.4.5.3).

### 6.4.4.2 Assumptions

The Earth point is specified in terms of $\phi$ and longitude $\lambda$ in radians, and altitude in meters. The look time need not be known but may be needed beforehand to convert any ECI or SC based vector to ECR. The input vector called "posECR" can be either a unit vector or a vector in meters, whose zenith and azimuth are desired, but in the case of the moon, it must be in meters.

*The normal will be defined by the point on the ellipsoid, ignoring altitude. Altitude will be used only as a pass–through to the refraction algorithm, and in calculating the parallax of the moon* The geometric effect of altitude on the angle of the CB vector is negligible except for the moon. There is no effect whatever of altitude on the Look vector direction when it is set by the instrument parameters in the spacecraft (for example, it if is obtained from PGS_CSC_GetFOV_Pixel(). ) Users wishing to supply their own ECR Look vectors based on terrestrial and spacecraft coordinates are responsible for taking the altitude into account.

The effect of the geoid in altering the zenith is also ignored; zenith means the normal to the ellipsoid.

Because of the way that geodetic latitude is defined, the calculations and algorithm herein are entirely independent of the Earth model except for the parallax correction, where WGS84 is assumed. Any difference in other models from WGS84 introduces negligible error (the error in angle is of order a few meters Earth radius difference divided by ~384,000,000 meters to the moon, i.e., about $10^{-9}$ radian or less than a milli arc second). The use of geodetic latitude $\phi$ as input guarantees that the rest of the algorithm is independent of Earth model, but it is the user's responsibility to provide latitude consistently with her/his other usage of Earth models.

### 6.4.4.3 Input Vector Tag

There are three different geometries of interest:

a. a vector to a celestial body at a very great distance, such as the sun, a planet, or a star

b. a vector to the moon and

c. the look vector (line of sight vector from Spacecraft to ground)

Classes (a) and (b) represent vectors pointing outward from Earth, but class (c) is a vector pointing inwards. The algorithm must allow for this difference, because the user expects the zenith angle to be between zero and 90 degrees (one $\pi/2$ radians) when the object (sun, moon, or star) is above the horizon or when the Earth point is visible to the spacecraft, respectively.

If this function is used to obtain the zenith and azimuth of the look vector, therefore, the vectorTag must be set to PGSd_LOOK to allow for the reversed sense of such a vector. If the zenith and azimuth of a distant celestial body (such as the sun or a planet) are desired, the user may supply PGSd_CB or any of the identifiers: PGSd_SUN, PGSd_MERCURY, PGSd_VENUS, PGSd_MARS, PGSd_JUPITER, PGSd_SATURN, PGSd_URANUS, PGSd_NEPTUNE, or PGSd_PLUTO. This is purely a convenience for users doing other calculations with a PGSd_CB identifier; the action of the function is in all cases the same—it finds the zenith and azimuth of the vector at the Earth point, without regard to parallax (i.e., the vector from Earth center to the Celestial body is regarded as unchanged due to the displacement of the look point from Earth center).

In the case of the PGSd_MOON, the geocentric parallax is appreciable, meaning that its apparent position is different as viewed from Earth center or from the look point. The difference can be as large as a degree. Therefore, in this case, a parallax correction is made, transforming the moon vector to topocentric origin. It is essential, in this case, of course, that the moon vector be supplied in meters. In this case, the input vector should be the Earth to moon vector defined from Earth center (geocentric), as obtained, for example, from the PGS_CBP_Earth_CB_Vector() tool. In all other cases, the input vector can be in any units, including normalized (unit vector).

There is obviously no parallax correction for the Look vector as it is already defined in ECR from spacecraft to Earth point, not to Earth center.

*Users wishing to take into account the minuscule parallax correction for the sun, or the correction for some other chosen body such as an asteroid, could simply label the vector as PGSd_MOON. (For the sun, the correction is only ~ 2.5 millidegrees.)*

#### 6.4.4.4 Refraction

Refraction by the atmosphere is calculated if the refraction flag is set to PGS_TRUE. This calculation approximately corrects, in the visual band, for the fact that any line of sight, such as the sun, moon, or look vector is bent by the atmosphere. In the current Toolkit, the atmosphere model was defective and the user was requested to limit use of the algorithm to under 8 km altitude, although the calculation was implemented to 20 km. In the present work and the next software increment, the algorithm has been remedied so that it works at any altitude, but it is hardly worth using over 15 km. When refraction is activated, the function also returns the decrease in zenith angle (in radians) to the user. Note that a *decrease in the zenith angle is reported as a positive number*, because the sign is always the same. This difference angle was slightly mis–reported in the current Toolkit, but the true refracted angle was accurate. See Section 6.5.5. For reference, the linear vector displacement of the ray and the changes in latitude and longitude are derived in Section 6.4.5.3, although the expressions are not currently implemented in a tool.

### 6.4.4.5 Line of Sight Below Horizon

If the vector is well below the horizon, a warning is returned and no azimuth calculation is done. The present algorithm is fairly forgiving for points slightly below the horizon (to 96 degrees zenith angle), so that the user interested in the location of the glow before sunrise or after sunset can find its azimuth; it is user responsibility to take special action between 90 degrees and 96 degrees if these data are not wanted.

The altitude is required only if refraction is to be calculated, and its only effect, apart from a tiny contribution to the lunar parallax, is to change the mean density of the atmosphere in the refraction function. Therefore the user wishing to have the refraction correction ought to supply the altitude off the geoid in meters. If refraction is not called for, the altitude can be set to zero (the correction in lunar parallax is much smaller than typical displacements of the lunar center of light from center of mass). The refracted zenith angle is substituted for the geometric one, but the actual change in refraction is also returned in this case.

If the zenith only flag is defined by the user to be PGS_TRUE the function will run faster but will not calculate the azimuth.

If desired, the unit look vector can be obtained in ECR by saving it from PGS_CSC_GetFOV_Pixel(); this will achieve very good performance.

If the azimuth is requested but the zenith angle is < 0.026 deg, it is deemed that the azimuth calculation is unreliable, because variations in the local vertical as determined from the geoid, and variable refraction in the atmosphere dominates at that level of accuracy. Zero azimuth and a warning notice are returned.

The azimuth is of no interest for day/night and twilight calculations. The flag for zenith only, normally PGS_FALSE, should be set to PGS_TRUE if it is desired to omit calculating the azimuth for speed. In that case the azimuth value is unpredictable.

### 6.4.4.6 Derivation of Algorithm

In the Lunar case the vector to the moon must be corrected for geocentric parallax.

To do this we first determine the flattening factor of the Earth from the WGS 84 axes A = 6378137.0 and C = 6356752.31414.

$$f = (A - C)/A \qquad (6.4\text{–}18)$$

We next calculate the ECR coordinates of the look point. (See Section 6.4.3.2 for the basis.) The altitude h is included here because it costs effectively nothing; but even at the top of the troposphere, the change in the angle of the moon does not exceed 6 seconds of arc. First, compute distance r of point from Earth's axis from the following two equations:

$$\xi = 1.0 / \mathrm{sqrt}(1.0 - (2.0*f - f*f) * \sin(\phi) * \sin(\phi)) \qquad (6.4\text{–}19)$$

$$r = ((A * \xi) + h) * \cos(\phi) \qquad (6.4\text{–}20)$$

Then get X,Y,Z at the Earth point

$X = r * \cos(\lambda)$ (6.4–21)

$Y = r * \sin(\lambda)$ (6.4–22)

$Z = ((A * (1.0-f) * (1.0-f) * \xi) + h) * \sin(\phi)$ (6.4–23)

Finally, the parallactic correction is made by subtracting (X,Y,Z) from the original CB position vector, posECR:

**posECR -> posECR - (X,Y,Z)**

In the case of the Look vector, posECR is simply reversed, so that it points towards the spacecraft. The vector  is then normalized. To show this it is denoted **normPosECR**.

The normal vector at the look point has ECR components:

$\mathbf{N}_X = \cos(\phi)\cos(\lambda)$

$\mathbf{N}_y = \cos(\phi)\sin(\lambda)$

$\mathbf{N}_Z = \sin(\phi)$

It is precisely because the use of geodetic latitude $\phi$ gives the true surface normal that this method works for any Earth ellipsoid, without knowledge of the particular axes or flattening.

The zenith angle is therefore

$\zeta$ = acos(**N** (*) **normPosECR**) where acos means arc–cosine and (*) means dot product

If refraction was requested, the latitude, altitude and zenith angle are passed to PGS_CSC_SpaceRefract(), which returns the refracted zenith angle. The latitude is not used in the current version, but it is hoped to implement it by the next increment, as the tropospheric density and scale height vary considerably with latitude.

If the azimuth $\psi$ is not required, the rest of the calculation is skipped. If $\zeta > 96$ deg (1.6755 radians) the sun is below the horizon and the azimuth is meaningless. In that case the function returns a warning message, sets $\psi = 0.0$ and returns.

To find the azimuth, define the projection of **normPosECR** on the Ellipsoid as

**nproj = normPosECR - (normPosECR (*) N)  N**

In the last term, the dot product in parentheses is a scalar multiplier for **N**. It is not necessary to normalize the vector **nproj** for the remaining operations; however, if the sun (moon) is at Zenith, it will be zero and the azimuth is indeterminate. Computationally, if we carry double precision, a number smaller than $10^{-11}$ or $10^{-12}$ may be regarded as zero (allowing for storage of exponents); but there will also be noise and error. Thus it was decided to consider 0.9999999 (seven 9's) to be a dot product so near unity that the sun is essentially at zenith. Therefore, if **normPosECR** (*) **N** > 0.9999999**,** the azimuth is set to zero (as it is indeterminate). In that case the remaining steps are omitted. The problem here is that **nproj** will be so short it is noise—a vertical post would cast no shadow and so you can't say if the shadow runs North or East!

Then we define North and East unit vectors on the ellipsoid by

$\mathbf{north}_X = -\sin(\phi)\cos(\lambda)$

$\mathbf{north}_Y = -\sin(\phi)\sin(\lambda)$

$\mathbf{north}_Z = \cos(\phi)$

$\mathbf{east}_X = -\sin(\lambda)$

$\mathbf{east}_Y = \cos(\lambda)$

$\mathbf{east}_Z = 0.0$

(these expressions were obtained by differentiating the normal vector with respect to latitude and longitude, respectively, and, in the latter case, re–normalizing because arc length along a small circle on the ellipsoid is not the differential of longitude).

the azimuth of the candidate vector is then:

$\psi = \mathrm{atan2}(\mathbf{east}\ (*)\ \mathbf{nproj}\ ,\ \mathbf{north}\ (*)\ \mathbf{nproj})$

which completes the algorithm.

### 6.4.5 Atmospheric Refraction for Space Observers

Function:

PGS_CSC_SpaceRefract()

Simple algorithms are developed for determining the angle of atmospheric refraction and the displacement of the point of Earth intersection due to refraction for space observing purposes. The methods obviate the need for finite difference calculations.

#### 6.4.5.1 Introduction

In space sensing of terrestrial data in optical wavelengths, atmospheric refraction affects the angle of Solar (or Lunar) illumination at the Earth, the viewing angle, and the actual position of the lookpoint. Rays of light striking the atmosphere are refracted toward the zenith as they descend, and outgoing rays away from the zenith as they ascend. In both cases, the ray is closer to the zenith lower in the atmosphere. For simplicity, then, the ensuing discussion will be in terms of an incoming ray, such as sunlight or moonlight, although the same calculation pertains to the outgoing rays used to observe the Earth from space.

The present work develops an analytic method that determines the angle of atmospheric refraction and the displacement of the point of Earth intersection due to refraction, accurate within about 1% as compared with a finite difference method. This compares well with the 10% or larger variations to be expected from the weather, which suggests that our four (4) equations can suffice for all but the most demanding applications. The four equations are: one rigorous conservation law; one geometric equation usually used to correct eclipse and occultation predictions for the heights of observatories; and two empirical formulas for refraction, spliced together at 83.9 degrees. A simple atmospheric model also allows extension to altitudes off sea level.

Existing studies and algorithms for refraction all seem to be derived from ground based measurements and expressed in terms of the refracted (Earth surface) zenith angle. This is natural, in view of the history of ground based optical observation. An Earth based observer sees a star at a certain zenith angle and wants to correct its position for comparison to the tabulated one. Algorithms that accept as their argument the observed, or surface zenith angle are unsuited for space observation, where the true zenith angle is known, and not the surface zenith angle. Thus, there is need for a simple approximation to refraction that accepts the true zenith angle as the argument. There is another difference in the geometry between ground based and space based observations. In the ground based case, the observer is occasionally interested in correcting eclipse predictions for the height of the observatory, and so is interested in the displacement of the refracted from the unrefracted ray *along the vertical*. In the case of space observation, the vertical displacement is of little interest, but the observer needs to know how far the true point of Earth contact of the ray is moved *along the horizontal* by refraction. The two problems are related.

At first glance, it might appear difficult to replace the various ad hoc recipes for refraction with one that depends on the angle of incidence; the structure of the atmosphere is complicated, and differential equations govern both the atmosphere and the propagation of the ray. It is not possible to assume plane geometry—a spherical atmosphere must be used. Indeed, MODIS has provided a brief FORTRAN finite difference program, originally written by Douglas Hoyt of Research Data Corporation (RDC), Inc., that has been used as a check on the present analytic work, while the *Supplement to the Astronomical Almanac* recommends iterating to solve for the refraction based on the incident angle. Yet it has been long recognized that there are certain conservation laws at work, even for a spherical atmosphere. See, for example, *A Manual of Spherical and Practical Astronomy*, by William Chauvenet (Lippincott, Philadelphia, 1885) pp. 129, 134–135 and pp. 515–516. It has been possible, by adding a little further analysis to Chauvenet's, to develop a set of equations that yield the zenith angle at the surface *analytically*, as a function of the unrefracted angle of incidence and the surface index of refraction only, while the displacement of the terrestrial point from the unrefracted one is found using empirical approximations. Thus, the extremely simple atmosphere model used here has only one purpose: to obtain the surface index of refraction; it is not used to integrate differential equations for the ray.

The philosophy in this work is that the refraction angle is substantial and important, but the displacement of the ray less so. Furthermore, the displacement is substantial only at large zenith angles, where it will be sensitive to weather effects, so there is no sense to try for high accuracy in the displacement using algorithms that do not make use of local weather data. It is fortunate that the empirical equations are needed only for the displacement, while the refraction angle is derived rigorously.

All the algorithms derived here will be for white light. Absorption limits the range of infrared and ultraviolet that one could include, and in the microwave and radio regions, the effects are more complicated, with contributions of positive sign from water vapor for microwaves and an index of refraction that can be less than unity for radio waves in the plasma regions. Atmospheric refraction is variable and sensitive to weather patterns. Only results for mean atmospheric conditions are treated here, and, initially, only for a globally averaged atmosphere, although it would not be difficult to add latitude dependence. We shall refer to the zenith angle at the top of the atmosphere above the observation point as the "true zenith angle," and that at the surface as the "refracted zenith angle." We ignore the oblateness of the Earth in this discussion, although it ought to be used in finding the Earth point of interest from the original space data. We also assume the zenith is defined by the ellipsoid and not the geoid. Table 6–2 gives the notation for the derivations, including the atmosphere model in Appendix D.

*Table 6–2.  Notation for Refraction and the Atmosphere Models*

| Symbol | Definition or value |
|---|---|
| A | Earth radius (meters) (spherical model) |
| h | altitude off the geoid, in meters, from observation point to the unrefracted ray |
| q | geocentric distance of any point along the ray |
| z | zenith angle in space, relative to the normal drawn from the base of the refracted ray (radians) |
| zr | running zenith angle of refracted ray relative to the Earth normal at the location beneath (radians) |
| z0 | zenith angle in space, relative to normal at base of *un*refracted ray (radians) |
| z' | zenith angle of refracted ray at the Earth's surface, relative to the local normal (radians) |
| Refr | conventional refraction angle in radians  = z - z' |
| Refr(z') | the same as a function of z' |
| Refr_deg | conventional refraction angle in degrees =decrease of zenith angle =180 * (z - z')/$\pi$ |
| Refr_arcSec | conventional refraction angle in seconds of arc |
| Refr0 | true refraction angle in radians = z0 - z' |
| dAng | angular displacement of the impact point ("footpoint") of the ray due to refraction (measured about Earth center, in radians) |
| d | linear displacement of the impact point ("footpoint") of the ray due to refraction (measured along Earth’s surface, in meters) |
| H | refracted ray's  elevation off horizon at Earth's surface, in degrees |
| $\mu$ | refractive index of air (a function of h) |
| $\mu 0$ | refractive index of air at the point of observation |
| T | temperature in Kelvins |
| SEA_TEMP | mean sea level temperature (Kelvin) = 288.115 K |
| tempFac | ratio of temperature to that at sea level: T/SEA_TEMP |
| TROPORATE | 0.0065 (degrees C temperature decrease per meter altitude increase) |
| TROPOPAUSE | height of tropopause = 10500 m |
| $\rho$ | density of the atmosphere (kg/m$^3$) |
| $\rho 0$ | density of the atmosphere (kg/m$^3$) at zero altitude, SEA_TEMP temperature |
| DENS_FAC | $\rho/\rho 0$ |
| R | ideal Gas Constant = 8314.3 J/k–mole |
| P | pressure (Pascals) (1 Pa = 1N/m$^2$ =  0.01 mb = 10 dyn/cm$^2$) |
| Pmb | pressure (mb) |
| g0 | 9.805 (mean sea level acceleration of gravity) |
| MEAN_MOLEC | 28.825 (see Appendix D) |
| W | density scale height, $\rho/[d(\rho)/dh]$ |
| $\Gamma$ | polytropic index such that $\rho \alpha T^{-\Gamma}$ in the troposphere |

Referring to Figure 6–4, note that there are three different angles that we must relate: z0 is the angle to the vertical at the intersection of the idealized unrefracted ray with Earth, z the zenith angle of the same (unrefracted) ray at the vertical of the true, or refracted, Earth intersection, and z' is the zenith angle of the refracted ray at the Earth's surface. In this discussion, "Earth's surface" means the final point of interest, be it on the ellipsoid, on some terrain, or on a low cloud layer. Note that the whole difference between z and z0 is due to the sphericity of the Earth, and that, as it descends, the ray also traverses laterally. Of course, the known quantity is z0, while that which is required is z'; the angle z is of interest for two reasons only: (1) the difference between z and z0 is needed to determine the displacement d, and (2) the original data on refraction were tabulated in terms of z' and fitted to a function that gave (z - z') from z’, without regard to z0. Indeed, the conventional angle of refraction is z - z’, but from the standpoint of space observation, the change in the zenith angle due to refraction is z0 - z'. Referring to the Figure R1, we see that z0 > z > z'. The difference between z0 and z is, however, quite small. Note that for very distant astronomical objects, there is a ray parallel to DP; which if unrefracted, would meet the Earth with zenith angle z (not z0) at P'. Because this ray could have been used in place of DP, it is quite reasonable that no distinction was made, traditionally, between z0 and z, except when discussing eclipses. Readers referring to Chauvenet's Fig. 44 should note that his point D is the same as ours, but we omit the line OD, which he draws, and we include the line OP, which he omits. Our usage of h, q, and μ is identical with his.

In the present context, we want to derive two quantities, dAng and z’, from z0, eliminating z. Three equations in the four variables dAng, z’, z, and z0 are needed, which will leave one degree of freedom, allowing for the independent variation of z0. We shall get the relation of z' and z from a standard refraction algorithm, and the two equations for dAng and z'(z0) from the theory of refraction in a spherical atmosphere. It will turn out that a *very simple relation between z' and z0 exists, independent of the details of atmospheric structure*, while to determine the displacement dAng requires knowledge of the angle z, whose value does depend on the structure and must be found from z' using empirical or semi–empirical formulas for z(z').

Two well–known analytic approximations for z(z') will be used for different ranges of the angles. The motivation for switching between these approximations will be discussed after the underlying theory. For the present, we refer to such algorithms generically as offering a function Refr(z') = z - z' as a function of z'.

### 6.4.5.2 Derivation of the Refraction Algorithm

Refer to the Figure 6.4. The unrefracted ray DP would have struck the Earth at P, making angle z0 with the vertical there in the absence of an atmosphere. It is refracted so as to strike the Earth at P', at an angle z' from the vertical there. The unrefracted ray meets the local vertical from P' at the angle z, which is usually denoted the unrefracted angle; because conventionally, the rays from a distant celestial object are parallel.

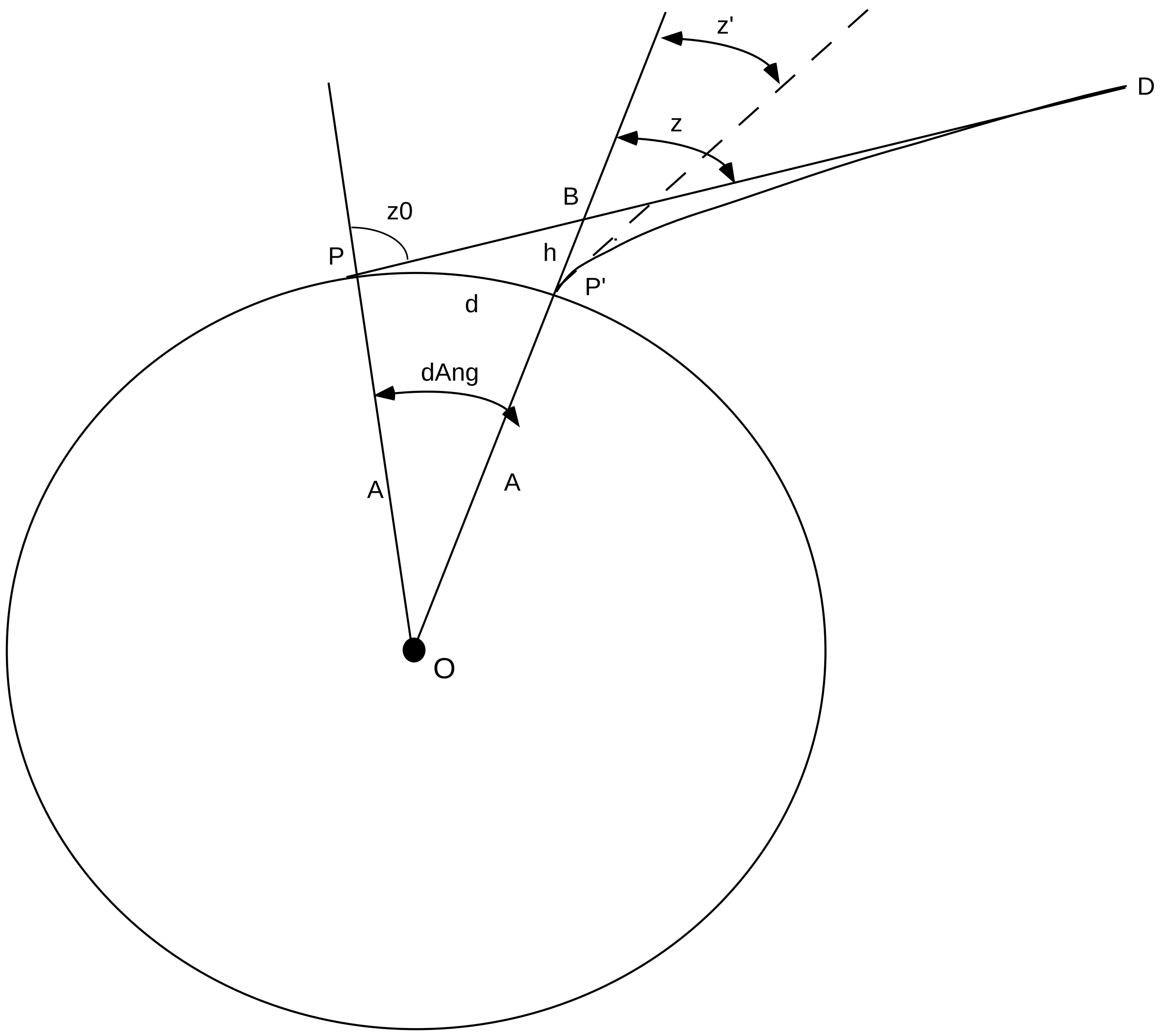


***Figure 6–4. Geometry of Terrestrial Refraction***

The (horizontal) displacement of the ray is in a vertical plane containing the ray and is in the sense that the actual (refracted) ray will meet the Earth d = dAng * A meters from the geometrical (unrefracted) position, on the side towards the nearest horizon. The angle "dAng" is the angle that the displacement in meters "d" subtends at Earth center.

From Chauvenet one finds two remarkable equations, rigorous for a spherical atmosphere,

$$q * \mu * \sin(zr) = \text{const} \qquad \text{(Chauvenet pp. 135, 516)}$$

which follows directly from Snell's law, and

$$1 + h/A = \mu 0 * \sin(z')/\sin(z) \qquad \text{(Chauvenet Equation 564, p. 516)}$$

In these equations, μ is the index of refraction at any altitude, and μ0 that at P'. The variable q is the geocentric distance, and will be used as a running variable along the ray, so that Chauvenet's relation q = A + h will not in general hold except at point B. The first equation will be shown to lead directly to an analytic relationship between z0 and z'. The second equation was originally intended to correct eclipse and occultation calculations for the height of the observatory, but it will be seen to solve the problem of determining dAng. The present problem is complicated by the fact that neither z nor z' is the known angle z0. If we can determine the difference z0 - z, we can solve triangle OBP for angle dAng = angle(POB) = angle(POP'), which is the radian equivalent of d. Summing angles around triangle POB and using the fact that z is the same as angle PBO, we see that

$$POP' + (pi- z0) + z = \pi \tag{6.4–23}$$

or

$$dAng = z0 - z. \tag{6.4–24}$$

Chauvenet's second equation looks temptingly as if we could use it to get z' from z or z from z', but in fact it does not contain enough information, which is why we need an empirical equation for z(z'). The first comes directly from Snell's law, but it also expresses a conservation law that we shall exploit. Remember that it deals with the angle between the ray and the *local* vertical—a vertical that swings round the Earth from OD to OP as the ray descends. Following up on this idea, we see that although it cannot give the refraction z - z', Chauvenet's first equation relates z0 and z'.

To derive the equation relating z0 and z', note that the first of Chauvenet's equations becomes that of a straight line in polar coordinates as q --> infinity, μ --> 1. Thus, outside the atmosphere, the equation is rigorous but seemingly of little interest. Yet it is most useful as a reference equation for *comparing* the relationship of zenith angle to geocentric distance along the unrefracted line DP and the refracted ray DP'. Along the straight line DP, which is the unrefracted ray, if zr0 is the running zenith angle (angle of the ray to the radius), then

$$q * \sin(zr0) = b \tag{6.4–25}$$

where b is a constant, namely the distance closest approach of the line DP to O. But from triangle OBP, because angle OPB has the same sine as its supplement, z0, we see that, applying the previous equation at P,

$$b = A \sin(z0) \tag{6.4–26}$$

Furthermore, by Chauvenet's first equation, with zr the running value of the zenith angle along DP', the actual ray has the equation

$$q*\mu* \sin(zr) = b1 \tag{6.4–27}$$

where b1 is another constant. But as q --> infinity, $\mu$ --> 1.0 and the values of zr are also the same on the refracted and unrefracted rays. Therefore,

$$b1 = b \qquad (6.4–28)$$

Therefore, sin(zr0) = $\mu$*sin(zr) when the comparison is at the same geocentric radius, so that q cancels.

Applying this equation at q = A, we see that

$$\sin(z0) = \mu 0 * \sin(z') \qquad (6.4–29)$$

precisely what is needed to close the system of equations. The remarkable power and simplicity of this equation makes one wonder why there are many complicated analyses (see, e.g., B. Garfinkel, *Astron. Journal*, Vol. **50**, pp. 169–179, (1944)) depending on atmospheric models and many fits depending on data. The reason is that for the classical case of terrestrial observation, z0 is not known, and precisely when the refraction becomes large, at large zenith angles, then the difference between z0 and z becomes significant. Remember, that even when the observed object is very far away, so that a ray parallel to DP will meet P' at angle z, it is only z that is then known from tabulated data on the star or planet; to know z0 one must also know d or dAng. The fact that Equation (6.4–29) seems previously unknown, although both Chauvenet and Garfinkel knew of Equation (6.4–27), may be largely due to their not having plotted point P and defined angle z0—an angle of interest mostly to space observers.

Chauvenet's second equation relates z and z0. This can be verified by using the law of sines on triangle OPB, yielding

$$(A+h)/A = \sin(z0)/\sin(z) \qquad (6.4–29)$$

which agrees with Chauvenet's second equation because sin(z') = $\mu$0 * sin(z0).

The problem, of course, is that z itself not known. Therefore, we first obtain z' by

$$z' = \mathrm{asin}(\sin(z0)/\mu 0) \qquad (6.4–30)$$

This gives the surface zenith angle, which defines the true angle of refraction as

$$Refr0 = z0 - z' \qquad (6.4–31)$$

To obtain d we need z, which must be obtained by applying some empirical equation. There is a wide choice—see the citations to the *Supplement to the Astronomical Almanac,* Chauvenet, and Garfunkel for example. A standard for astronomers has been: *Astrophysical Quantities, 3rd Ed.* by C.W. Allen, (London, the Athlone Press, 1973) Hereinafter equations from this book are referenced as “Allen” equations with a page citation.

The problem received attention from such well–known astronomers as Bradley, Argelander, and Bessel. At this point, it is best to apply a little common sense to the selection of such an equation, remembering that variations in the weather can cause differences of a few percent at small zenith angle to perhaps 100% near the horizon. The touchstone of an equation at large zenith angle will be its smoothness and its reputation for fitting the data well near the horizon. The touchstone at small zenith angle should be quite different. We shall develop analytical results that constrain the equations for small z, where, in any case, the refraction tends to zero, so that fractional accuracy of the measurements is less and the fitting to the data is less reliable. Equation (3.283–1) from the *Explanatory Supplement to the Astronomical Almanac* gives a sea level refraction angle that passes through zero at a zenith angle of about 0.0775 degrees, and a negative answer for the refraction at z = 0. (The equation is only offered as an approximation, and is obviously not intended to be used at zero zenith angle where everyone knows the refraction is zero—but for the present program we need a better treatment near zero zenith angle because a poor approximation yields an implausible displacement there.) Knowing the correct analytic behavior for Refr at z --> 0 is important so that the displacement d behave reasonably near the zenith. We thus derive an approximate analytic form for z - z' as z --> 0.

It is not difficult to show that at z0 --> 0, the difference in z and z0 must tend to 0; but it is possible to say more than that. For small zenith angles, the Earth curvature is less important, so we can estimate d using a plane atmosphere approximation. If we assume a single homogeneous layer, whose thickness is one density scale height W, Figure 6.5 shows that d ~ W (z-z'). Even though we get the linear dimension d from a plane parallel model, for such small angles we can still interpret it in the spherical Earth context, using Equation 6.4–24, as d = A * (z0 - z).

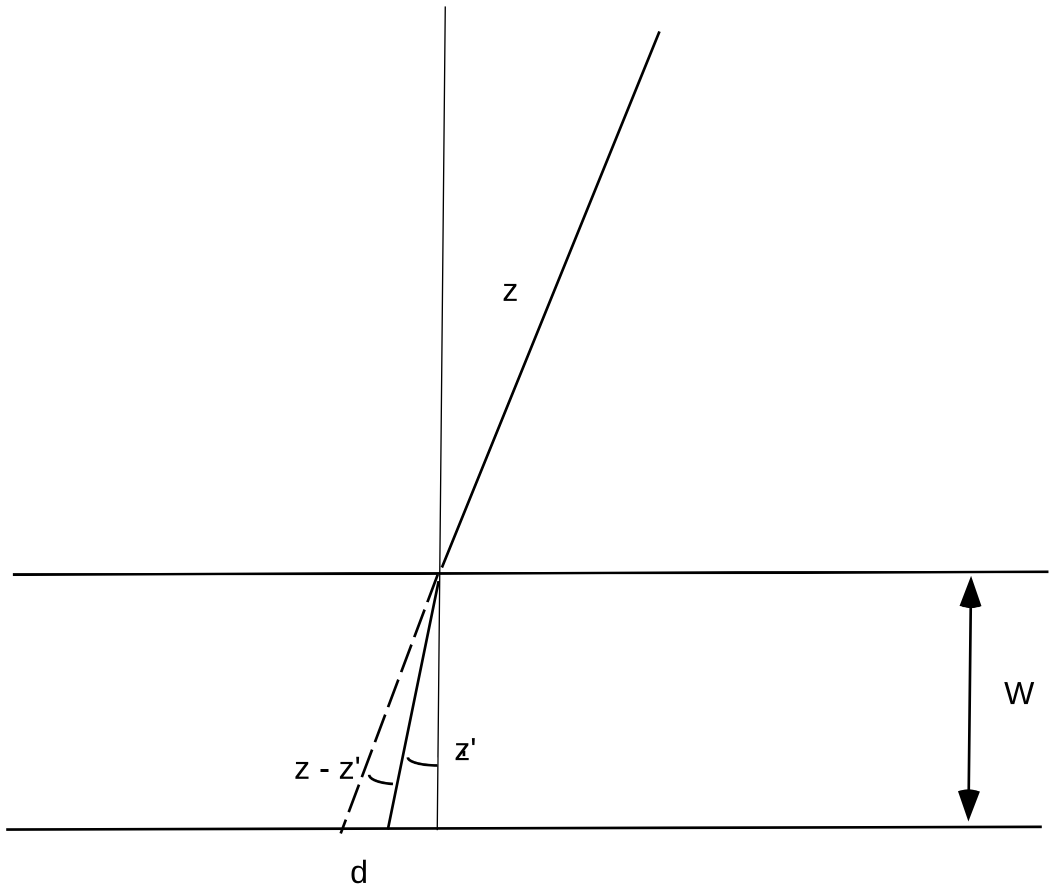


***Figure 6–5. Estimated Refraction Near the Zenith***

Thus,

For $z \to 0$, $z0 - z \sim (W/A) * (z - z')$ (6.4–32)

Although this equation is only approximate, depending as it does on an assumed oversimplified atmospheric model, it is good enough for small zenith angles and it gives a good picture of why the difference in z0 and z, although small, is so significant. Also, it enables us to calculate analytically an approximate formula for z - z'. To do this we apply Equation (6.4–30) in the limit of small z and z0, to yield $z0 \sim \mu 0 * z'$. Combining with Equation (6.4–32), we see that

For $z \to 0$, $Refr = \{(\mu 0 - 1)/[1 + (W/A)]\} * z'$ (6.4–33)

It is now possible to derive analytically the leading constant in Allen's (1962) approximation

$Refr\_arcSec = 58.3 \tan(z') - 0.067 \tan^3(z')$ (A1) (Allen, p. 124)

Using a mean sea level index of refraction 1.0002904, (from Hoyt's model in the MODIS software) A = 6371000 m (the mean Earth radius), and W = 8970 m, we find from Equation (6.4–33):

For $z \to 0$, Refr = 0.0002899 radians * z' = 59.52 arc seconds * z' (6.4–34)

with z', as usual, in radians. Using $\tan(z') \sim z'$ and ignoring the cubed term, we see that Allen's 58.3 arc seconds is explained as the value of $(\mu 0 - 1)/[1 + (W/A)]$ for a slightly smaller index of refraction than ours. We thus choose Allen's approximation for small zenith angles, changing the 58.3 to 59.52. For large zenith angles, the standard algorithm to relate z' and z from the *1992 Explanatory Supplement to the Astronomical Almanac* (U.S. Naval Observatory) on p. 144, Equation (3.283–1), was used. [Equation (3.283.2) was compared and found to offer no advantage.] Finally, to make a smooth transition at z = degrees (rad), it was necessary to change Allen's coefficient of $\tan^3(z')$ very slightly. The final equation used for angles less than 1.465 radians (83.94 degrees) was

$Refr = \{(\mu - 1.0)/[1 + (W/A)]\} * [\tan(z') - 0.00117 * \tan^3(z')]$ (6.4–35)

It is to be emphasized that the derivation in Equations (6.4–32)–(6.4–34) is only to validate Allen's constant and to assure continuity with Equation (3.283–1) of the Supplement; for example, a factor $\sec^2 z$ is omitted from (6.4–32) on the grounds that z is small. The overall effect is that Allen's numerical constant can be replaced consistently by the factor $(\mu - 1.0)/[1 + (W/A)]$, so that the dependence on altitude or on the assumed baseline index of refraction, $\mu$, is consistent, but the peculiar dependence on the tangent and its cube cannot be derived this way.

For larger angle, we use an equation adapted from Equation (3.283–1), which reads:

$Refr\_deg = 0.0167 \text{ degrees} * [(0.28*Pmb)/T] / \{\tan[(H + 7.31/(H + 4.4)]\}$ (AAS 3.283–1)

The problems requiring some minor adaptation are that H is the elevation of the ray at surface level, $H = (180/\pi) * (\pi - z')$ (deg), and that Pmb/T is really a complicated representation of the air density. This form was no doubt chosen because pressure and temperature are easily measured, while density is not. To an excellent approximation, 0.28*Pmb/T is unity at sea level for T = SEA_TEMP, so we'll replace it with

$$0.28*Pmb/T \dashrightarrow \rho/\rho0 \qquad (6.4\text{–}36)$$

The value of ρ/ρ0 is taken from an atmospheric model—see Appendix D. Finally, when the zenith angle z is obtained from one of the two empirical equations, then dAng is found from Equation (6.4–24), which completes the solution, since d = A * dAng. In Toolkit 3, equation (6.4–35) was not yet known. Use of Equation (AAS 3.283–1) in its place resulted in excessive and unreasonable values for dAng near the zenith, so an ad hoc reduction factor was implemented.

In the Toolkit 3, the refraction angle was returned as the conventional one, z - z', for consistency with the usual refraction equation in the A.A. supplement, and the refracted zenith angle returned to the calling program was z0 - (z - z'), not z'. Even though z0 and z are very nearly equal (much closer to each other than z'), it is correct to return z' as the refracted zenith angle and the angle of refraction as z0 - z'. This change has been made in Toolkit 4.

### 6.4.5.3 Horizontal Displacement

Finally, the equations are given for transforming dAng to changes in latitude and longitude. To find the change in latitude and longitude, the user can use the following:

The trace of the ray path on the Earth is a vector lying ψ radians East of North, where ψ is the azimuth. Therefore, the displacement of the footpoint of the ray is

***Table 6–3. Vector Displacement on Surface***

| Direction | Value |
|---|---|
| North | d* cos(ψ) |
| East | d* sin (ψ) |

where d = dAng* A. Thus, the increments in latitude and longitude are

***Table 6–4. Displacement in Latitude and Longitude***

| Direction | Value |
|---|---|
| latitude (ϕ) | dAng * cos(ψ) |
| longitude (λ) | dAng * sin(ψ)/cos(ϕ) |

The latter expression is singular at the North and South poles and the user should avoid using it there. The displacement of the ray can be assumed to be South at the North pole and North at the South pole but when starting at either pole, the longitude (not its increment) must be found from atan2(yray,xray) where (xray,yray,zray) are the components of the look vector in ECR. After calling PGS_CSC_SpaceRefract(), then, the user who is interested in the displacement in latitude and longitude needs to implement these equations. The Toolkit software does not perform these operations, but it is hoped to add the functionality in a later release.

#### 6.4.5.4 Sample Results

The following table exemplifies results at sea level, using a conversion of 6378137 m per radian on the displacement.

*Table 6–5. Refraction Results at Sea Level*

| Zenith Angle In Space (deg) | Zenith Angle At Surface (deg) | Refraction (deg) | Linear Displacement (meters) |
|---|---|---|---|
| 10.000000 | 9.997066 | 0.002934 | 0.549 |
| 20.000000 | 19.993944 | 0.006056 | 1.223 |
| 30.000000 | 29.990394 | 0.009606 | 2.221 |
| 40.000000 | 39.986039 | 0.013961 | 3.983 |
| 45.000000 | 44.983363 | 0.016637 | 5.464 |
| 50.000000 | 49.980174 | 0.019826 | 7.725 |
| 55.000000 | 54.976243 | 0.023757 | 11.399 |
| 60.000000 | 59.971192 | 0.028808 | 17.846 |
| 61.000000 | 60.969996 | 0.030004 | 19.697 |
| 62.000000 | 61.968722 | 0.031278 | 21.816 |
| 63.000000 | 62.967361 | 0.032639 | 24.257 |
| 64.000000 | 63.965905 | 0.034095 | 27.080 |
| 65.000000 | 64.964340 | 0.035660 | 30.366 |
| 70.000000 | 69.954333 | 0.045667 | 58.380 |
| 75.000000 | 74.938025 | 0.061975 | 136.072 |
| 76.000000 | 75.933417 | 0.066583 | 166.728 |
| 77.000000 | 76.928121 | 0.071879 | 207.384 |
| 78.000000 | 77.921967 | 0.078033 | 262.469 |
| 79.000000 | 78.914723 | 0.085277 | 338.977 |
| 80.000000 | 79.906069 | 0.093931 | 448.379 |
| 81.000000 | 80.895543 | 0.104457 | 610.332 |
| 82.000000 | 81.882461 | 0.117539 | 860.316 |
| 83.000000 | 82.865762 | 0.134238 | 1266.534 |
| 84.000000 | 83.843713 | 0.156287 | 1970.638 |
| 85.000000 | 84.813286 | 0.186714 | 2974.066 |
| 86.000000 | 85.768718 | 0.231282 | 4858.394 |
| 87.000000 | 86.697712 | 0.302288 | 8677.416 |
| 88.000000 | 87.569758 | 0.430242 | 17538.457 |
| 89.000000 | 88.295108 | 0.704892 | 41818.325 |
| 90.000000 | 88.619113 | 1.380887 | 113429.256 |

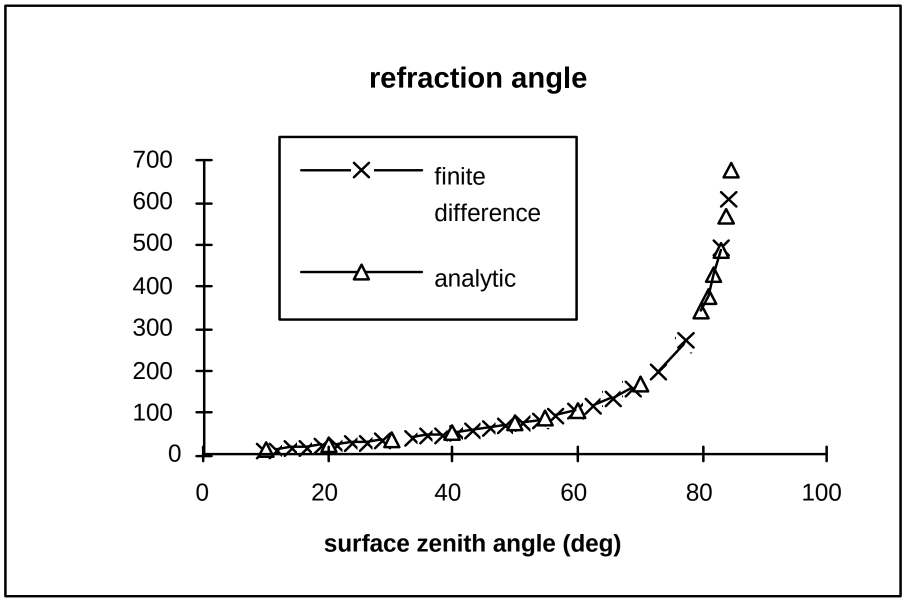
Note that the linear displacement at 88 degrees zenith angle is almost 17.5 km—very substantial. Because of the very approximate atmosphere model, this number could vary by perhaps 25% depending on weather in temperate and tropical regions; in the Arctic it would be considerably smaller. Comfortingly, however, at an incident zenith angle as large as 85.25 degrees (85.05 degrees refracted), our result for the displacement, 3.33 km, agrees with the result from Hoyt's finite difference program (2.91 km) within 15%. The displacement at 90 degrees incidence, over 113 km, is only suggestive and could easily vary by 50%.

The composition of the atmosphere in Toolkit 3 was obtained from Allen's "Astrophysical Quantities, 3rd ed." (London, the Athlone Press, 1973) p. 121, because the U.S. Standard Atmosphere (National Oceanic and Atmospheric Administration (NOAA), 1976) is dry, which seemed unrealistic. The effect on the molecular weight is small, but the even larger effect of water vapor on the lapse rate is ignored here. The atmosphere model is used only to get the index of refraction at sea level. Latitude dependence is not implemented in the present  version. Later, the sea level temperature and mean scale height will be altered to become functions of latitude.

(If the zenith angle in space exceeds 90 degrees; algorithm fails and returns with error condition.)

#### 6.4.5.5 Validation

The refraction angle and displacement were compared with Hoyt’s finite difference program at sea level, 10 km, and 20 km altitude. The results at sea level are shown on the following figures. The results at altitude are nearly as close.



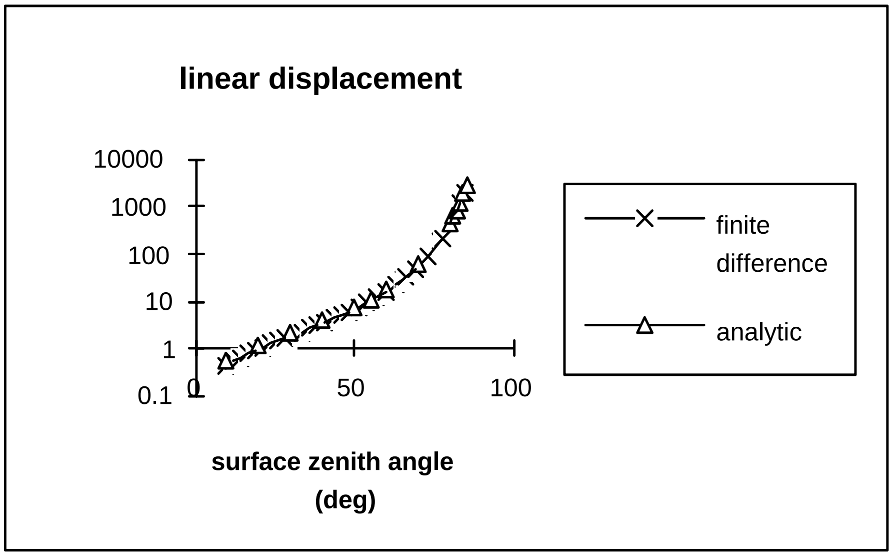


***Figure 6–6. Sea Level Results Compared to Hoyt's***

### 6.4.6 Transformation Between Orbital and SC

Function:

PGS_CSC_ECItoORBquat()

The orbital coordinate system is shown in Figure 6–6.

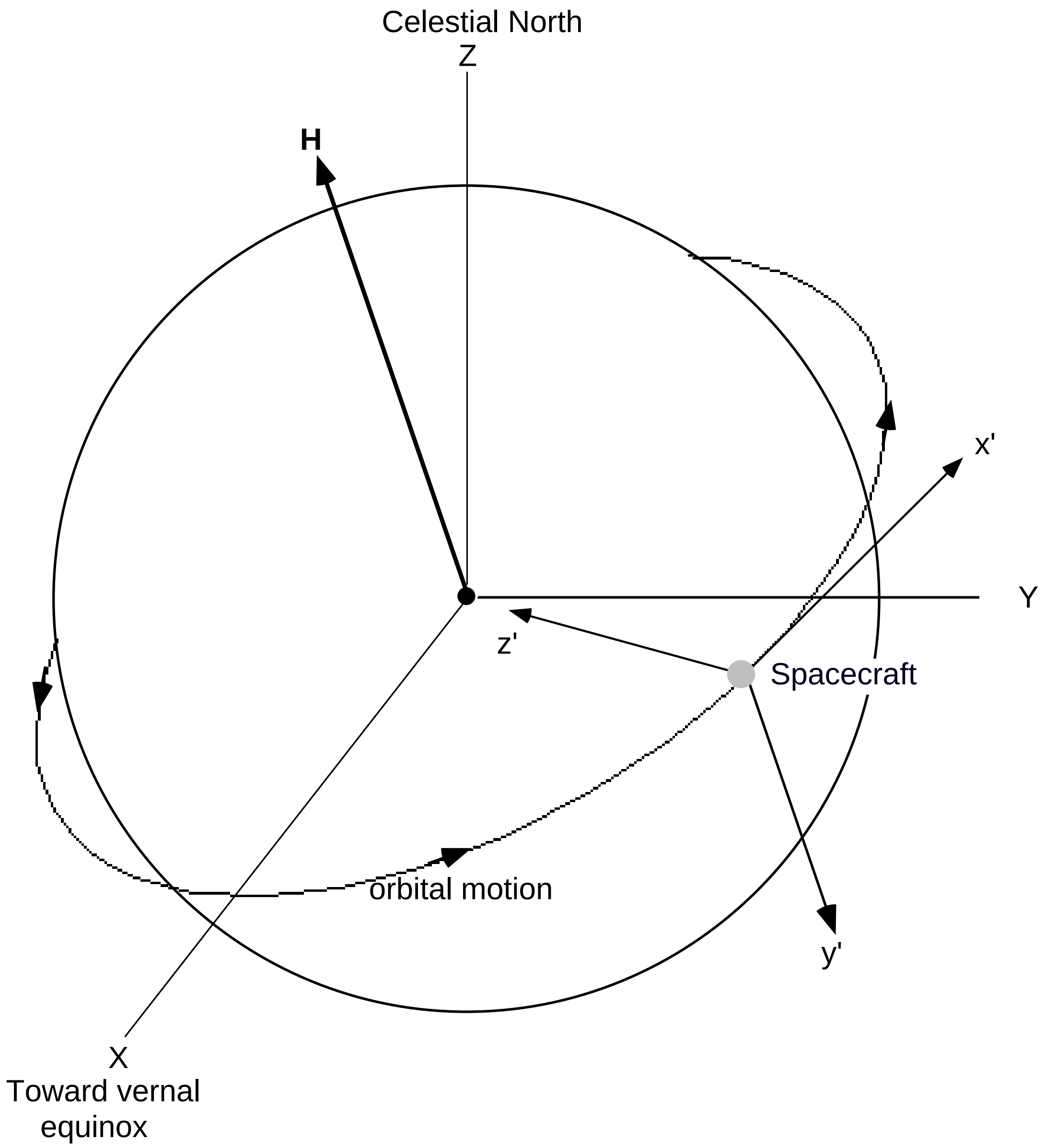


***Figure 6–7. The Orbital Coordinate System as Related to J2000 ECI Coordinates***

In this figure, the J2000 coordinates are denoted (X,Y,Z) and the Orbital (x',y',z'). The y' axis is normal to the orbit, anti–parallel to the angular momentum **H**, the z' axis is towards geocentric nadir, and the x' axis is along y' × z' where × is the vector cross product. The rotation matrix from ECI to orbital coordinates is constructed as follows: The ECI position vector **R** is normalized and reversed to yield $\mathbf{e}_Z$. Then the ECI velocity **V** is orthogonalized to $\mathbf{e}_Z$ by subtracting off its component along $\mathbf{e}_Z$ and then it is normalized to yield $\mathbf{e}_X$. Next the cross product of $\mathbf{e}_Z$ and $\mathbf{e}_X$ is taken to form $\mathbf{e}_Y$. Then the attitude matrix to transform from Orbital to ECI is formed by making its columns the three unit vectors just constructed. The attitude matrix is converted to a quaternion in the function PGS_CSC_getQuats(), and then its spatial components are reversed so that it will go from ECI to Orbital. This methodology enabled the direct importation of UARS code. In Toolkit 5 the three unit vectors will be put into the rows of the matrix, and no reversal of the final space components of the quaternion will be performed.

The same function is applied in the TOD system instead of in J2000 for the case of TRMM, because TRMM is referenced to TOD. After the quaternion is derived, the TRMM ephemeris is put into J2000.

This page intentionally left blank.

# 7. Pixel and Sub Satellite Point Tools

## 7.1 Introduction

Tools:

- PGS_CSC_GetFOV_Pixel()
- PGS_CSC_SubSatellitePoint()

These two tools are discussed together because both depend on the assumed Earth figure, both involve intimately the Satellite's relationship to the Earth, and they deal with fairly simple geometry. The tools in Section 8 deal with the complications of an arbitrarily shaped field of view.

### 7.1.1 Organization

This section is organized as follows:

7.1—Introduction

7.2—Suggested Usage

7.3—PGS_CSC_GetFOV_Pixel()

7.4—Subsatellite Point

7.5—The Day/Night Indicator

### 7.1.2 Summary

The tool PGS_CSC_GetFOV_Pixel locates pixels on the Earth. It also delivers the ECR pixel vector for further processing, and the slant range and range rate. The slant range can be used for scaling purposes or to obtain light travel time, and the range rate for Doppler work.

The tools PGS_CBP_body_inFOV() and PGS_CSC_Earthpt_FOV() accept a FOV definition in terms of its perimeter and tell the user if a candidate Earth point or any portion of a celestial body is in the FOV.

## 7.2 Suggested Usage

A detail description of the usage of this tool is provided in the *SDP Toolkit Users Guide for the ECS Project*

## 7.3 PGS_CSC_GetFOV_Pixel()

This tool is really a pixel by pixel geolocation tool, although a whole array of pixels can be processed in one invocation. First an outline is given. Then any points needing elaboration are covered in detail.

### 7.3.1 Outline

The calculations are done in a carefully staged way. First the look vector is normalized and transformed from spacecraft to ECI coordinates. Then it is corrected for aberration (and normalized again). The actions after that depend on the accuracy flag. It set "normal" a quick check for Earth intersection is done, and if satisfied, the spacecraft position, velocity, and the instrument look vector are all transformed to ECR, after which the intersection, slant range, and Doppler velocity along the line of sight are all calculated. If the accuracy flag is set "high" (PGS_TRUE) several more actions are taken. First, when the preliminary determination of Earth intersection is performed, the slant range is estimated, and the ECI to ECR transformation is computed at the time when light left the lookpoint, instead of at the observation time. See column three of Tables 7–1 and 7–2 for the size of the effect. (The spacecraft position and velocity are, of course, still carried in ECI at the observation time before transforming to ECR; only the Earth orientation is affected.) Also, before the SC to ECI transformation, and before calculating the intersection, the spacecraft position is displaced by the user–supplied amount of the instrument offset (this is done with a scratch vector, so the ephemeris itself is not affected). If the offset exceeds 120 meters in length, it is assumed that an error was made (perhaps in units), the warning message PGSCSC_E_INSTRUMENT_OFF_BOARD is issued and that time offset is not processed further. The offsets must be specified for each time offset when the high accuracy mode is set; this allows data for different instruments on different booms, or an instrument on a slewing boom, to be interspersed.

It should be noted that the ECI to ECR transformation used here not only transforms position in the obvious way, but also transforms the velocity vector, allowing both for the geometric part (change of coordinate system) and for the increment to velocity in going to a rotating reference frame (cross product of radius vector and Earth rotation vector).

### 7.3.2 Aberration of the Unit Look Vector

The spacecraft is traveling at a vector velocity $\mathbf{v} = (v_x, v_y, v_z)$ in ECI coordinates[6]. In this frame a LEO spacecraft is going at roughly 6900 m/s, or $2.3 \times 10^{-5}$ $c$, where $c$ is the speed of light. As a result of the spacecraft motion, any light ray seen by the spacecraft is aberrated so that it seems to come more from the forward direction. Ref: *Explanatory Supplement to the Astronomical Almanac*, p. 129. The vector **p**' representing a unit vector **p** as seen by the moving observer is found from

[6] Of course, the ECI coordinate system is not strictly inertial, because its origin moves with the Earth., subject to acceleration by the Sun and Moon. For our present purposes, that system can be regarded as inertial, however, with the Sun's and Moon's gravity represented as tidal terms if needed.

**p**' = [**p**+**v**/**c**]/| [**p**+**v**/**c**] | (7.3–1)

In the present case, the problem is that we observe **p**' and want **p**. Therefore, it is necessary to reverse the sign in the equation and interchange **p** with **p**'. That is what is encoded in the function in the PGS toolkit. The astute reader will have noticed that the operation of reversing the sign and interchanging **p** with **p**' does not lead to a closed set of equations. In other words, if we vectorially solve Equation (7.3–1) for **p** we don't get the same answer as by interchanging **p** with **p**' and reversing **v**. There is no real problem, because the differences are second order in v/c, but the equation is only accurate to first order in v/c.

Some surprise having been expressed that this correction is significant; it is worth estimating the size of the effect. Consider a flat Earth approximation, and assume all significant vectors lie in a vertical plane. (This approach is only for illustrative purposes, Equation (7.3–1) is actually implemented, and it does not depend on a flat Earth approximation.)

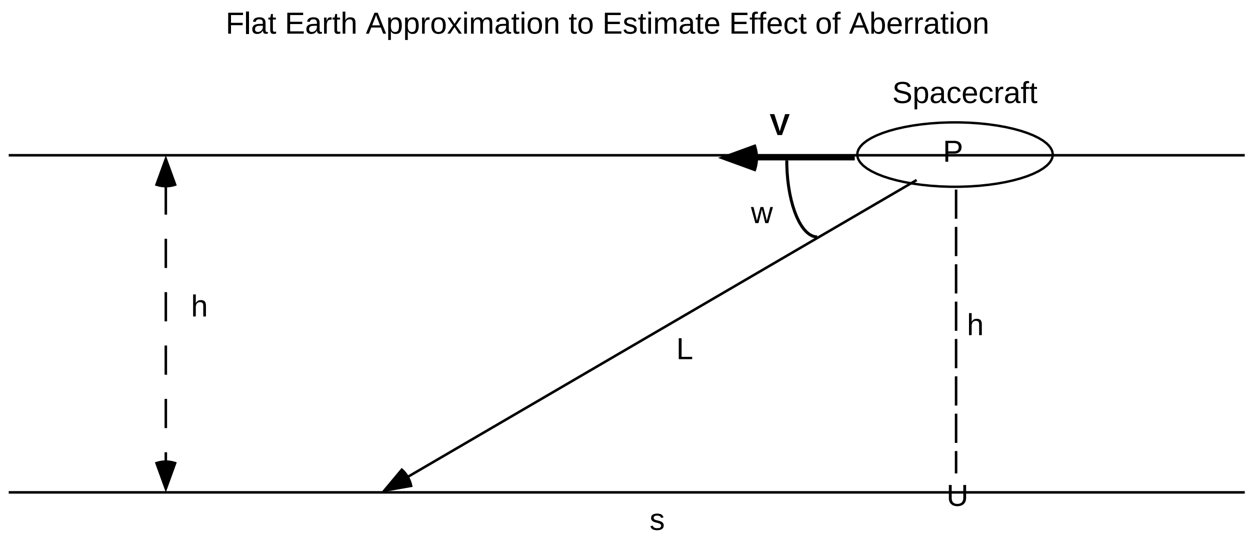


***Figure 7–1. Flat Earth Approximation to Estimate Effect of Aberration***

In the diagram, let the angle of the look vector L make the angle w with the velocity **V** in the *Earth* reference frame. The look vector means, in this context, the apparent line of travel of a light ray or photon. **V** is assumed horizontal. Let the altitude be h. Then the distance from Nadir U to the lookpoint is

s = h cot(w) (7.3–2)

The effect of aberration is that the angle w in the spacecraft frame is less by the amount

dw = (V/c) sin(w) (7.3–3)

Thus, when the sensor is nominally pointed at a certain angle from V, in the spacecraft reference frame, it sees a photon that makes a larger angle with V in ECI. That is why the PGS tool *subtracts* V/c from the unit look vector; the sensor is actually looking further from V. The apparent value of s is thus s+ds where

$$ds = h\, d[\cot(w)] = h\,(dw)\,[\csc^2(w)] \qquad (7.3\text{–}4)$$

combining the last two equations, we find

$$ds = h\,(V/c)\,\csc(w) = \text{slantRange} * (V/c) \qquad (7.3\text{–}5)$$

For EOS AM1, the nadir angle can be as large as 64 degrees. *w* is the complement of the nadir angle *n* (in this plane model). Thus, for *h* = 705 km, w = 26 deg, *V* = 6900 m/s, we find *s* = 37 m. This is the along–track error for the plane, flat Earth model. The error for looking cross track would be of order the slant range times dw. In actual SDP Toolkit operation, of course, no plane nor flat Earth approximation is used. In that case, while aberrating the look vector is correct, to obtain an analytic or geometric derivation of the geolocation change due to aberration is a bit more daunting. Clearly, however, there are cases (with the look vector toward the fore) where a look vector would miss intersecting the Earth disk if it were not corrected for aberration, while to the aft, there are directions near the limb where the look vector when aberrated correctly fails to meet the Earth disk, while if unaberrated it would do so. This correction is much larger than that for Earth rotation (generally less than 3 m) so it is always made.

### 7.3.3 Coarse Calculation Of Geolocation Contact, Slant Range, and Angle Of Incidence

The purpose of this section is to show how one can determine if the line of sight meets the Earth, and, if so, the slant range and the zenith angle at contact. This is a simplified calculation based on a spherical Earth model circumscribed around the true, oblate Earth. It is used only for set–up and time saving features; accurate zenith angle is available in the tool PGS_CSC_ZenithAzimuth() and the accurate slant range is calculated with the function PGS_CSC_LookPoint() when the preliminary calculation now being discussed finds that Earth intersection is likely. The time savings with this method accrues because a quick determination of Earth intersection, not dependent on Earth orientation, but only on altitude and nadir angle, saves the user the cost of invoking the ECI to ECR transformation. The rapid determination of slant range permits correction for Earth rotation in *advance of* determining the lookpoint. This calculation is performed only when the accuracy flag is set high, and it avoids iterative invocation of the subfunction that calculates the actual lookpoint and the accurate slant range. It is exercised after the aberration correction.

The zenith angle at contact is not used in the function, PGS_CSC_GetFOV_Pixel(), but it is calculated in this section because so many instrument teams express their scans or attitude adjustments in terms of the zenith angle of the look vector. The calculation herein is an approximate one based on a spherical Earth. It refers to Figure 7–2.

Let the spacecraft at point P be h meters above the Earth, whose center lies at O. Suppose that the unit look vector **N** is along the line PQ, making angle n to the nadir vector PO, and passing closest to O at Q, where the distance of the projected look vector to Earth center is q. Let the total distance OP be denoted Rs. Assume a spherical Earth model, with radius A equal to the semi–major Earth axis. In that case:

$$Rs = A + h$$

and the look vector will intersect the spherical Earth model if it intersects the (inscribed) ellipsoidal Earth model. Given A, h, and N it is desired to know if an Earth intersection W exists, and if so, what is the slant range L and what is the zenith angle $\theta$ of the look vector N at W. By using the semi–major axis we will occasionally (really rarely) call for the calculation of the lookpoint when it does not exist, because the line of sight misses the Earth narrowly, at high latitude, when it would intersect a spherical Earth circumscribed about the true one. In that case, the function PGS_CSC_LookPoint() will report no intersection, so no harm is done. The latter function works with the true Earth ellipse.

### 7.3.4 Approximate Earth Intersection

To determine approximately if an Earth intersection exists, envision the limiting case where the look vector is tangent to the Earth, so that points W and Q coincide, and q = A. From the triangle PWO (which has a right angle at W in that case), we see that

$$\sin(n) = A/Rs \quad \text{(limiting case of tangency at Earth limb)} \qquad (7.3–6)$$

An intersection would then seemingly exist if $\sin(n) < A/Rs$. Unfortunately, this is not the case, and the equation must be used in a slightly different form to test for Earth intersection. The use of the sine function lets an alias intersection occur with spacecraft zenith and nadir interchanged. In other words, the sine of the supplement of the nadir angle n is the same as the sine of n, so if the spacecraft rolled 180 degrees, or the look vector pointed away from Earth (as it will occasionally for Clouds and Earth's Radiant Energy System (CERES)), a false positive test would ensue. Thus, the inverse sine function is applied to the equation, so it reads

$$n < \text{arcsine}(A/Rs) \qquad \text{implies Earth intersection occurs}$$

The branches of the arc sine function in the "C" function library suffice to eliminate the alias solution.

### 7.3.5 Slant Range:

Next we want to get the slant range, L in terms of h (or Rs) and n. Refer to the diagram for the geometry.

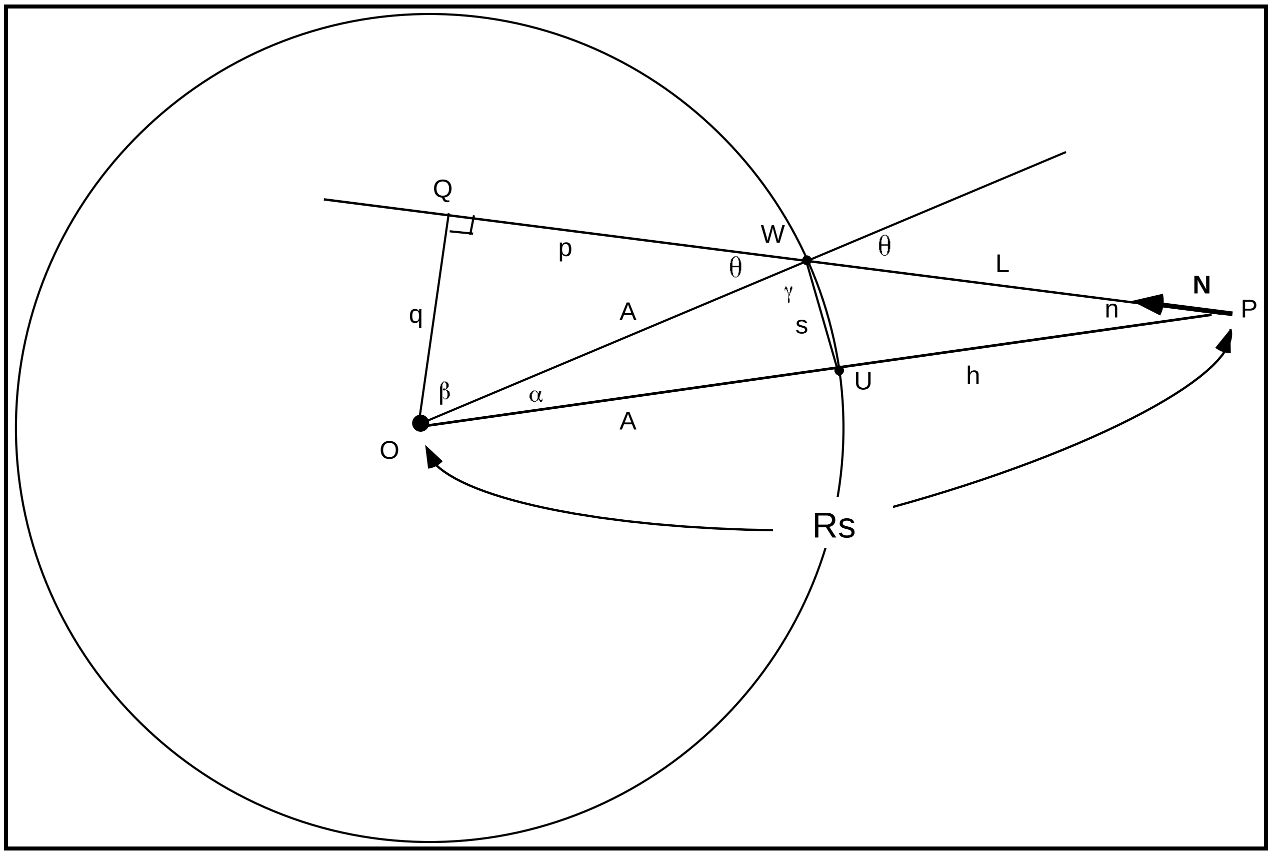


***Figure 7–2. Slant Range***

Let p be the distance QW and

$$D = p + L \qquad (7.3–7)$$

Then from the triangle OQP, we have

$$p = [A^2 - (A + h)^2 + D^2]^{1/2} = [D^2 - 2\,A\,h - h^2]^{1/2} \qquad (7.3–8)$$

From the same triangle, we find

$$D = Rs \cos n \qquad (7.3–9)$$

On eliminating p in the form p = D - L, we find

$$L = Rs \cos(n) - \text{sqrt}\{[(Rs \cos(n)]^2 - 2\,A\,h - h^2\} \qquad (7.3–10)$$

which is coded in the algorithm. This is an approximate slant range suitable for estimating light travel time. It is used in the high accuracy case only. In all cases, the accurate slant range is calculated later in the function PGS_CSC_LookPoint(). In the Toolkit 3 implementation, the mean Earth radius was used for Rs. Because it is performed before calling PGS_CSC_LookPoint(), the approximate slant range calculation can encounter an erroneous negative argument in the square root in Equation (7.3–10) when the line of sight (look vector) misses the Earth narrowly, such that it would intersect the circumscribed sphere. In Toolkit 4 the equatorial radius is used for Rs, preventing the problem at negligible loss of accuracy. The only loss of accuracy is the change in Earth rotation based on the slightly incorrect slant range, and is less than a millimeter. The messaging was improved in the Toolkit 4 version to indicate that the line of sight narrowly misses the Earth in the special case just explained. Of course, many instruments routinely slew past Earth limb, and for these, the user will probably wish to ignore the warning return and message, “PGSCSC_W_MISS_EARTH,” which merely says that the look vector did not intersect the ellipsoid. Tables 7–1 and 7–2 show the typical slant ranges, zenith angle at the look point, and the Earth motion at the equator during the light travel time for the TRMM, AM and PM spacecraft.

***Table 7–1. TRMM Slant Range, Unrefracted Zenith Angle and Earth Motion During Light Travel , for Altitude 351.26 km***

| **Nadir Angle** | **Slant Range** | **Earth Motion** | **Zenith Angle At Look Point** |
|---|---|---|---|
| (deg) | (km) | (m) | (deg) |
| 0.00 | 351.3 | 0.54 | 0.0 |
| 2.50 | 351.6 | 0.55 | 2.6 |
| 5.00 | 352.7 | 0.55 | 5.3 |
| 7.50 | 354.5 | 0.55 | 7.9 |
| 10.00 | 357.0 | 0.55 | 10.6 |
| 12.50 | 360.3 | 0.56 | 13.2 |
| 15.00 | 364.4 | 0.57 | 15.8 |
| 17.50 | 369.3 | 0.57 | 18.5 |
| 20.00 | 375.2 | 0.58 | 21.2 |
| 22.50 | 382.0 | 0.59 | 23.8 |
| 25.00 | 389.9 | 0.60 | 26.5 |
| 27.50 | 399.0 | 0.62 | 29.2 |
| 30.00 | 409.4 | 0.64 | 31.8 |
| 32.50 | 421.2 | 0.65 | 34.5 |
| 35.00 | 434.8 | 0.67 | 37.2 |
| 37.50 | 450.2 | 0.70 | 40.0 |
| 40.00 | 467.8 | 0.73 | 42.7 |
| 42.50 | 488.0 | 0.76 | 45.5 |
| 45.00 | 511.3 | 0.79 | 48.2 |
| 47.50 | 538.2 | 0.83 | 51.1 |
| 50.00 | 569.7 | 0.88 | 53.9 |
| 52.50 | 606.9 | 0.94 | 56.8 |
| 55.00 | 651.4 | 1.01 | 59.8 |
| 57.50 | 705.5 | 1.09 | 62.9 |
| 60.00 | 773.0 | 1.20 | 66.0 |
| 62.50 | 859.8 | 1.33 | 69.4 |
| 65.00 | 977.4 | 1.52 | 73.0 |
| 67.50 | 1151.3 | 1.79 | 77.1 |
| 70.00 | 1469.0 | 2.28 | 82.5 |
| 71.00 | 1748.4 | 2.71 | 86.0 |
| 71.38 | 2037.0 | 3.16 | 89.0 |

***Table 7–2. AM1 and PM Slant Range, Unrefracted Zenith Angle and Earth Motion During Light Travel for Altitude 705 km***

| Nadir Angle | Slant Range | Earth Motion | Zenith Angle At Look Point |
|---|---|---|---|
| (deg) | (km) | (m) | (deg) |
| 0.00 | 705.0 | 1.09 | 0.0 |
| 2.50 | 705.7 | 1.09 | 2.8 |
| 5.00 | 708.0 | 1.10 | 5.6 |
| 7.50 | 711.8 | 1.10 | 8.3 |
| 10.00 | 717.1 | 1.11 | 11.1 |
| 12.50 | 724.1 | 1.12 | 13.9 |
| 15.00 | 732.8 | 1.14 | 16.7 |
| 17.50 | 743.3 | 1.15 | 19.5 |
| 20.00 | 755.8 | 1.17 | 22.3 |
| 22.50 | 770.5 | 1.20 | 25.1 |
| 25.00 | 787.5 | 1.22 | 28.0 |
| 27.50 | 807.1 | 1.25 | 30.8 |
| 30.00 | 829.7 | 1.29 | 33.7 |
| 32.50 | 855.6 | 1.33 | 36.6 |
| 35.00 | 885.4 | 1.37 | 39.6 |
| 37.50 | 919.7 | 1.43 | 42.5 |
| 40.00 | 959.3 | 1.49 | 45.5 |
| 42.50 | 1005.4 | 1.56 | 48.6 |
| 45.00 | 1059.5 | 1.64 | 51.7 |
| 47.50 | 1123.5 | 1.74 | 55.0 |
| 50.00 | 1200.5 | 1.86 | 58.3 |
| 52.50 | 1294.9 | 2.01 | 61.8 |
| 55.00 | 1414.0 | 2.19 | 65.5 |
| 57.50 | 1571.1 | 2.44 | 69.5 |
| 60.00 | 1794.4 | 2.78 | 74.1 |
| 62.50 | 2172.1 | 3.37 | 80.1 |
| 63.00 | 2293.5 | 3.56 | 81.7 |
| 64.00 | 2716.3 | 4.21 | 86.5 |
| 64.20 | 2972.9 | 4.61 | 89.0 |

The preliminary slant range is then used to estimate the light travel time in the high accuracy case, and the ECI to ECR transformation of the SC ephemeris is done at the pre–dated time.

### 7.3.6 True ECR Intersection

Function:

PGS_CSC_LookPoint( )

The function PGS_CSC_LookPoint() is now invoked to determine the ECR rectangular coordinates of the intersection. The equation for the ellipsoid

$$X^2/A^2 + Y^2/B^2 + Z^2/C^2 = 1 \qquad (7.3–11)$$

with B = A

and the vector equation defining a line along the look vector:

$$X\_vec = X\_SC + L * lookVec \qquad (7.3–12)$$

where **X_vec** = (X,Y,Z)

can be combined into a single quadratic equation for the slant range L. When there is an intersection, there are two real roots and the smaller yields the visible intersection. When there is tangency, there is only one root, which is a valid solution. When the line of sight misses the Earth, the roots are complex and are not calculated; the warning PGSCSC_W_MISS_EARTH is returned. This condition will be quite common for scanning instruments that scan past Earth limb so the number of messages written to the logfile was limited to 25 per invocation.

### 7.3.7 Further Processing

The rectangular coordinates from PGS_CSC_LookPoint() are converted to longitude $\lambda$ and latitude $\phi$ with the equations

$$\lambda = \mathrm{atan2}(Y,X) \qquad (7.3–13)$$

$$\phi = \mathrm{atan2}(Z,(1\text{-}ecc^2)*\mathrm{sqrt}(X^2 + Y^2)) \qquad (7.3–14)$$

where ecc is the eccentricity of the Earth ellipsoid, Equation (6.4–1). (See P. R. Escobal, *Methods of Orbit Determination*, Wiley, N.Y. p. 28, equations 1.58, 1.59.) This transformation was incorrect in Toolkit 3 resulting in a latitude error of up to 5.8 arc minutes.

## 7.4 Subsatellite Point

This function finds the subsatellite point and its velocity. First, the latitude and longitude of point on the terrestrial spheroid directly beneath a spacecraft of known position and space velocity are found from the ECR to geodetic transformation, PGS_CSC_ECRtoGEO(). An algorithm developed herein is then used for determining the North and East components of the velocity of this point as well as the rate of change of altitude. Refer to Figure 7.3.

### 7.4.1 Introduction

Consider a spacecraft at position (*x*,*y*,*z*) in rectangular coordinates, with velocity (*xdot*,*ydot*,*zdot*). The Earth ellipsoid (technically a spheroid in the approximation that it is a figure of revolution about the North to South or z axis) has semi–major axis A (the equatorial axis) and semi–minor axis *B*= *A* sqrt(1-ecc$^2$) where ecc is the eccentricity of the spheroid. The spacecraft has at any time a "subsatellite point" Psub defined by dropping a normal to the spheroid. Earth scientists in the EOS observing program have expressed the desire to know the velocity Vsub of the point Psub, i.e., the terrestrial velocity of the "ground track." This paper presents a readily computable algorithm for determining that velocity. The algorithm was coded and tested and is part of the EOSDIS geolocation package. Although there was no request for the rate of change of altitude, it falls out of the analysis with little extra effort. This quantity could be compared to a radar altimeter if one were used, while the other two velocity components only represent motion of a geometric point across the surface of the Earth. The function PGS_CSC_SubSatPoint() first finds the latitude and longitude of the subsatellite point by accessing the spacecraft ephemeris for its coordinates and velocity, and transforming these to ECR coordinates. It then uses PGS_CSC_ECRtoGEO() to obtain the latitude, longitude, and altitude of the satellite, which are inputs for the more difficult calculations presented here. Note that the velocity of the subsatellite point is not the projection of the spacecraft velocity on the Earth's surface, or on the local tangent plane to the Earth's surface. This is easily seen, for example, by considering a hypothetical orbit of zero inclination. Examining the situation from a perspective along the polar axis, one sees that while the satellite executes in one orbit a circle whose radius is its distance from the Earth's center, the subsatellite point executes a smaller one, whose radius is the equatorial radius of the Earth. Thus, the velocity of the subsatellite point is reduced in proportion to the ratio of the latter to the former.

Subsatellite Point and its Velocity

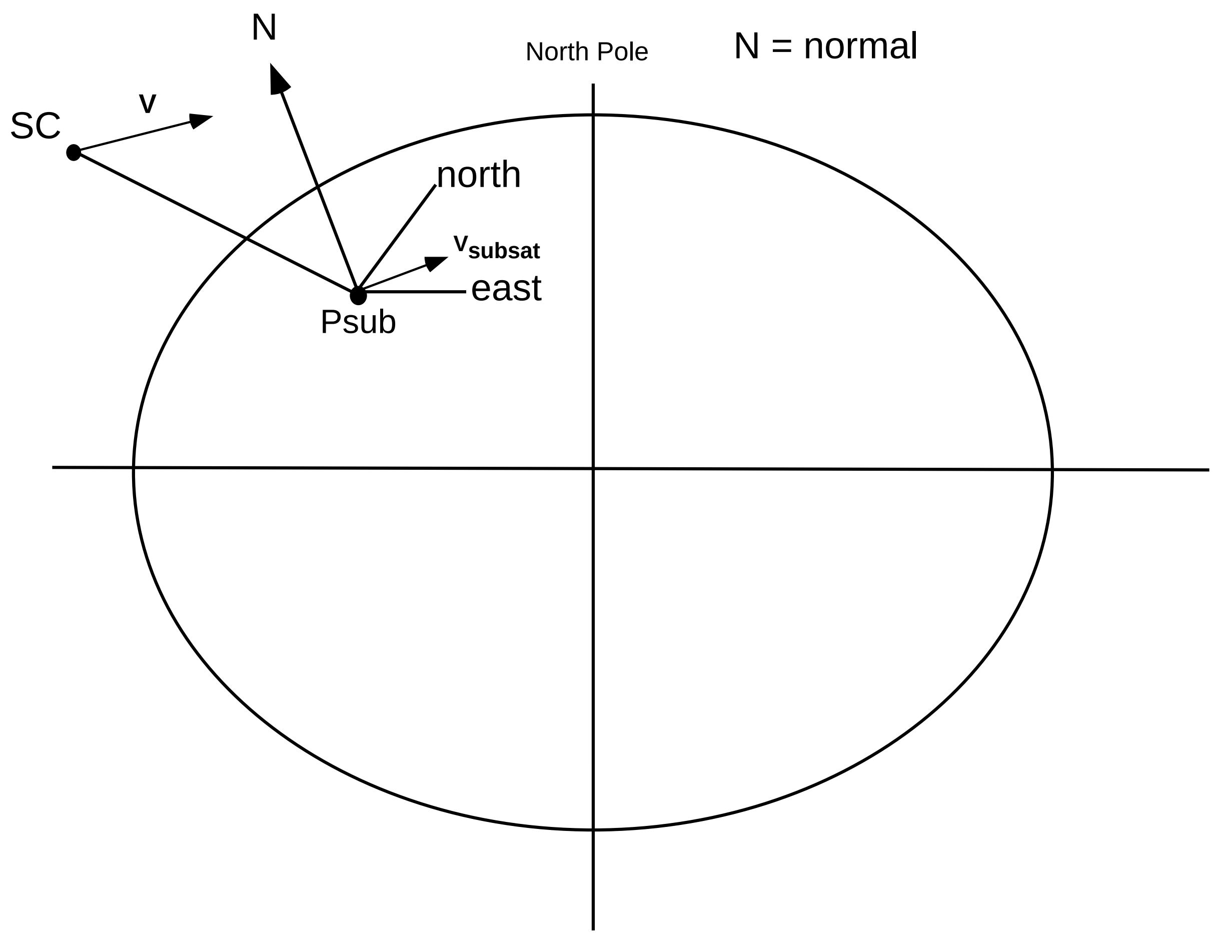


***Figure 7–3.  Subsatellite Point and its Velocity***

The analysis here is all based on an *instantaneous, geometric model, with no allowance for aberration or light travel time*. An instrument on the spacecraft looking directly at nadir will find a spot slightly aft of nadir, and also the position of the spot very slightly affected by Earth rotation during the light travel time (assuming that the ECI to ECR transformation of spacecraft position is done at the instant of observation). These effects can be assessed with the tool PGS_CSC_GetFOV_Pixel(), which allows for aberration always and for both effects in high accuracy mode.

### 7.4.2 Analysis

The point Psub lies at a certain latitude $\phi$ and longitude $\lambda$. Note that while $\lambda$ is identical with the usual azimuthal coordinate $\lambda$ of spherical coordinates; $\phi$ is defined differently in two ways. First, as the latitude rather than the co–latitude, $\phi$ runs from $-\pi/2$ at the South Pole, through 0.0 at the equator, to $+\pi/2$ at the North pole, where $\pi$ is the ratio of the circumference of a circle to its diameter. (The co–latitude runs from 0 at the North Pole to $\pi$ at the South Pole; the notation using a Greek "$\phi$" for latitude is common in spacecraft engineering; see Escobal, 1965 for example[7].) The second difference between $\phi$ and a common spherical coordinate is that it is defined as the angle between the normal to the spheroid and the equatorial plane. This normal does not in general pass through the center of the spheroid. The "geometric" or "astronomical" latitude is the angle between a vector from the origin to the point of interest and the equatorial plane. It is the complement of the usual spherical polar coordinate. The latitude we use here is technically the "geodetic latitude," because in geodesy the local normal is extended to the sky where its tip locates a point whose declination on the celestial sphere is equal to the selfsame geodetic latitude.

If *h* is the altitude of the spacecraft the transformation from coordinates ($\phi$,$\lambda$,h) to (x,y,z) is:

$$x = \cos(\phi)[A*\xi +h]\cos(\lambda) \qquad (7.4–1)$$

$$y = \cos(\phi)[A*\xi +h]\sin(\lambda) \qquad (7.4.2)$$

$$z= \sin(\phi)[A*\xi(1-ecc^2) + h] \qquad (7.4–3)$$

where

$$\xi = 1.0/\mathrm{sqrt}[1.0 - ecc^2 (\sin(\phi))^2] \qquad (7.4–4)$$

Note that the ($\phi$,$\lambda$,h) coordinate system is left handed.

The transformation (7.4–1:4) can be inverted analytically only in the case h=0, or by the use of the complex solution of the quartic equation, which is cumbersome. Any of several simple iterative techniques suffice to invert it to the accuracy required for practical spacecraft work in a few iterations. (For example, one such scheme gives answers good to 5 millimeters out of about 7,000 km in 2 to 9 iterations, for all values of $\phi$ and $\lambda$ when h lies between -1,000 m and + 10,000 km.) Such methods solve the problem of determining the point Psub; for simple reason that once $\phi$ and $\lambda$ are known at the spacecraft; they are known at Psub. In other words, if the spacecraft is at ($\phi$,$\lambda$,h) then Psub is ($\phi$,$\lambda$,0).

[7] P. R. Escobal *Methods of Orbit Determination* (J. Wiley, New York) 1965

The Toolkit has a requirement to supply velocity Vsub of the point Psub. To determine that velocity is a more trying exercise than finding Psub itself. The problem is worked here in the Earth Centered Rotating reference frame, where the North Pole is the z axis, a line from Earth center in the equatorial plane toward the Prime Meridian (that of Greenwich, England) is the x axis, and the y axis forms an orthogonal right handed triad. See the figure for details.

Define s as arc length on the spheroid, and let the rates of change of $\phi$ and $\lambda$ be denoted

$d(\phi)/dt = d\phi\_dt; d(\lambda)/dt=d\lambda\_dt$ (7.4–5)

respectively. Then the velocity has North component

$vn = [ds/d(\phi)]d\phi\_dt$ (7.4–6)

and East component

$ve = [ds/d(\lambda)]d\lambda\_dt$ (7.4–7)

It is easily shown that if (xp,yp,zp) is the subsatellite point in ECR rectangular coordinates, then

$ds/d(\phi) = A(1\text{-}ecc^2)\xi^3$ (7.4–8)

and

$ds/d(\lambda) = \text{sqrt}(xp^2 + yp^2)$ (7.4–9)

It only remains to determine the rates of change of $\phi$ and $\lambda$ as defined in Equation (7.4–5). These are found from the chain rule

$d\phi\_dt = d\phi\_dx*xdot + d\phi\_dy*ydot + d\phi\_dz*zdot$ (7.4–10)

and

$d\lambda\_dt = d\lambda\_dx*xdot + d\lambda\_dy*ydot + d\lambda\_dz*zdot$ (7.4–11)

also the rate of change of altitude can be found from

$dh\_dt = dh\_dx*xdot + dh\_dy*ydot + dh\_dz*zdot$ (7.4–12)

where, for example, d$\phi$_dy refers to the partial derivative of $\phi$ with respect to y (at constant x and z), etc., and the satellite's velocity in ECR coordinates is (xdot,ydot,zdot). The problem now reduces to finding the partial derivatives in Equations. (7.4–10:12) from Equations (7.4–1:4). While it is, in principle, possible to use the solution of the quartic, the reader may recall that this solution contains canceling imaginary expressions, and that the choice of branches is needed in effecting the cancellation. Since we do not need the solution per se, it will suffice, instead, to differentiate Equations (7.4–1:4) for the partial derivatives of (x,y,z) with respect to ($\phi$,$\lambda$,h) and then to invert the resulting matrix. The answer will be the matrix of partial derivatives of ($\phi$,$\lambda$,h) with respect to (x,y,z), i.e., the Jacobian. (K. G. J. Jacobi, *De Formatione et Proprietatibus Determinentium*, Konigsberg, 1841; or see D.V. Widder, *Advanced Calculus*, Prentice–Hall, New York 1947.)

The elements of the matrix to be inverted are (in the same notation)

$$J(1,1) = dx\_d\phi = - [A*\xi^3*(1-ecc^2) + h]\sin(\phi)\cos(\lambda) \quad (7.4–13)$$

$$J(1,2) = dx\_d\lambda = - [A*\xi + h]\cos(\phi)\sin(\lambda) \quad (7.4–14)$$

$$J(1,3) = dx\_dh = \cos(\phi) \cos(\lambda) \quad (7.4–15)$$

$$J(2,1) = dy\_d\phi = - [A*\xi^3*(1-ecc^2) + h]\sin(\phi)\sin(\lambda) \quad (7.4–16)$$

$$J(2,2) = dy\_d\lambda = [A*\xi + h]\cos(\phi)\cos(\lambda) \quad (7.4–17)$$

$$J(2,3) = dy\_dh = \cos(\phi)\sin(\lambda) \quad (7.4–18)$$

$$J(3,1) = dz\_d\phi = [A*\xi^3*(1-ecc^2) + h]\cos(\phi) \quad (7.4–19)$$

$$J(3,2) = dz\_d\lambda = 0 \quad (7.4–20)$$

and

$$J(3,3) = dz\_dh = \sin(\phi) \quad (7.4–21)$$

It is not difficult to show that the determinant of J is

$$\Delta = - \cos(\phi)[A^2*\xi^4 *(1-ecc^2) + A*\xi*h + A*h*\xi^3*(1-ecc^2) + h^2] \quad (7.4–22)$$

where the sign is always negative due to the left handed nature of ($\phi,\lambda$,h). The solution for the partials needed in Equations (10–11), with the abbreviation

$$\Psi = 1.0/\Delta \quad (7.4–23)$$

and the notation K for the inverse of matrix J is:

$$d\phi\_dx = K(1,1) = \Psi*(A*\xi+h)\cos(\phi)\sin(\phi)\cos(\lambda) \quad (7.4–24)$$

$$d\phi\_dy = K(1,2) = \Psi*(A*\xi+h)\cos(\phi)\sin(\phi)\sin(\lambda) \quad (7.4–25)$$

$$d\phi\_dz = K(1,3) = - \Psi*(A*\xi+h) * [\cos(\phi)]^2 \quad (7.4–26)$$

$$d\lambda\_dx = K(2,1) = \Psi*[A*\xi^3*(1-ecc^2)+h]\sin(\lambda) \quad (7.4–27)$$

$$d\lambda\_dy = K(2,2) = - \Psi*[A*\xi^3*(1-ecc^2)+h]\cos(\lambda) \quad (7.4–28)$$

$$d\lambda\_dz = K(2,3) = 0 \quad (7.4–29)$$

Notice that Equations (26–27) lead to a singularity at the poles because $\Psi$ tends to infinity there and no factor cos($\phi$) is present to compensate. This singularity is trapped and reported by the algorithm.

The remaining elements of K—the partials of h with respect to (x,y,z) are used to determine hdot, the rate of change of the altitude. Thus, they are:

$$dh\_dx = K(3,1) = - \Psi*[A*\xi+h] [A*\xi^3*(1-ecc^2)+h][\cos(\phi)]^2*\cos(\lambda) \quad (7.4–30)$$

$$dh\_dy = K(3,2) = - \Psi*[A*\xi+h] [A*\xi^3*(1-ecc^2)+h][\cos(\phi)]^2*\sin(\lambda) \quad (7.4–31)$$

$$dh\_dz = K(3,3) = -\Psi*[A*\xi+h]\ [A*\xi^3*(1-ecc^2)+h]\cos(\phi)*\sin(\phi) \qquad (7.4\text{–}32)$$

These are substituted in Equation (7.4–12) to give the rate of change of altitude.

### 7.4.3 Conclusions And Further Developments

The foregoing algorithm seems the most effective way to calculate the required velocity. It was coded in C and tested on several UNIX workstations at Hughes ECS Development Facility (EDF) facility. The algorithm is part of the Toolkit geolocation package. Several new algorithms were developed for the preliminary task of finding the subsatellite point itself. Some of these are readily generalized to the triaxial case. It is hoped to report on these algorithms and to extend the present analysis for that more general case.

The author is indebted to David Withoff of Wolfram Research for checking the equations.

## 7.5 The Day/Night Indicator

Tool:

PGS_CSC_DayNight()

This tool determined day, twilight, and night conditions at any user selected latitude, longitude, and times by comparing the unrefracted Solar zenith angle with certain standard angles selected by the user by a sunZenithLimitTag. The tool invokes PGS_CBP_EarthCB_Vector() to get the sun vector, which is then put into ECR with PGS_CSC_ECItoEECR(). Then PGS_CSC_ZenithAzimuth() yields the Solar zenith angle. The altitude is taken to be zero. The user supplied tag can be any of the entries in the left column, except Day, because the tests are all against the limits in the right hand column. The comparison values are taken from the *1995 Astronomical Almanac* as follows:

***Table 7–3. Limiting Sun Zenith Angles for Twilight and Night***

| Test Condition | Least Sun Zenith Angle |
|---|---|
| PGSd_CivilTwilight | 90 deg 50 minutes (1.5853407 radians) |
| PGSd_CivilNight | 96 degrees (1.6755161 radians) |
| PGSd_NauticalNight | 102 degrees (1.7802358 radians) |
| PGSd_AstronNight | 108 degrees (1.8849556 radians) |

the PGSt_boolean value afterDark is returned as true if the zenith angle exceeds the limiting value, falseif equal or less. Because the listed values allow for average atmospheric effects, no refraction correction is used. There is no correction for terrain or altitude.

# 8. Earth Point or Celestial Body in FOV Tools

## 8.1 Introduction

Tools:

- PGS_CSC_Earthpt_FOV()
- PGS_CBP_body_inFOV()

The two tools PGS_CSC_Earthpt_FOV() and PGS_CBP_body_inFOV() have similar purposes and designs. In both cases, they determine if something is in the FOV. In the former case it is an Earth point of user–specified longitude, latitude and altitude. In the latter case it is a user specified celestial body, or any part of that body, or it can be a user specified point in ECI coordinates. In any case, the FOV must first be defined by the user. Because the two tools share certain geometric problems they are discussed together. Since the computations to determine a definitive answer can be lengthy, a large number of cases are excluded by a “conical hull” test, which can quickly ascertain if the Earth point or CB is far from the field of view.

### 8.1.1 Organization

This section is organized as follows:

### 8.1.2 Summary and Overview

Here it is explained how the two tools resemble each other and differ, and then the reader is directed to the relevant sections.

#### 8.1.2.1 Similarities and Differences in the Two Tools

The two tools discussed here have the following in common:

- they both require the field of view to be specified the same way in SC coordinates
- they both have to solve the problem of determining whether a given point is inside a given perimeter

- they both must check for Earth blockage of the line of sight
- they both return a Boolean flag to indicate object in or not in the FOV
- they both return the vector to the candidate point in SC coordinates. This convenience avoids the user's having to call again for information already found within the tool, and enables the user to check, if desired, on the exact relationship of the point(s) of interest and the FOV.

The two tools differ, however in these ways:

- PGS_CSC_Earthpt_FOV() takes latitude, longitude and altitude as inputs, while PGS_CBP_body_inFOV() requires a Celestial Body tag as input. (In the generic case "PGSd_STAR," the latter tool accepts any point defined by the user in ECI.)
- PGS_CSC_Earthpt_FOV() works with a point only, while PGS_CBP_body_inFOV() uses a finite radius body, except in the generic case, PGSd_STAR, when a user defined ECI point is supplied. In the finite radius case, complicated algorithms must be used to find any overlap of a disc representing the body and the FOV.
- In PGS_CSC_Earthpt_FOV() we must check that the point is not occulted by the *near* portion (~ hemisphere) of the Earth, while for PGS_CBP_body_inFOV() we must check for blockage by any part of the Earth. This, in the first case, the tool will report "PGS_TRUE" when and only when the required Earth point is on the near side and in the FOV aperture, while in the second case the tool must report "PGS_FALSE" if the periphery of the Earth blocks the FOV or occults the object, or a combination of these two effects.

#### 8.1.2.2 How to Read the Remaining Sections

The common items between the two tools are discussed first, in Sections 8.2–8.4, following which separate sections consider the different problems of the two. The reader concerned with PGS_CSC_Earthpt_FOV() should read Sections 8.2, 8.3, and 8.5, while the reader concerned with PGS_CBP_body_inFOV() should read Sections 8.2, 8.3, and 8.6.

## 8.2 FOV Definition

The user is required to supply the FOV specification in terms of nFOV "perimeter vectors" $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$....,.$\mathbf{x}_p$, that must delimit the perimeter, and that *must* be in order—clockwise or counter–clockwise[8], where

[8]If the perimeter vectors are supplied out of order, the resulting FOV will be tested haphazardly as if it were valid, with unpredictable results. For example, a square FOV with perimeter points ABCD listed in the order ACBD is a bowtie shaped figure. By the definitions in Section 8.3, the two triangular lobes and the exterior will be treated such that the portion containing the **inFOV** vector is the "inside" and everything else "outside."

p = nFOV - 1

The tips of the vectors are assumed to be connected by arcs of great circles. Thus, the FOV is considered to be a polygon on the sky, or on a large sphere around the instrument and spacecraft, just as the sky is considered a large sphere by terrestrial observers. To permit some simplifications of the algebra (using Euclidean vector operations rather than spherical trigonometry) *it is assumed that each perimeter vector makes an acute angle with the **inFOV** vector*. The user also supplies another vector **inFOV** vector, which is in the FOV and is requested to be near the center (if not, the speed of the algorithm will deteriorate). This specification must be supplied for each time at which the tools are called, which allows for slewing or scanning instruments. Although the tools only say whether the item is in the FOV or not, both also return the vector to the point in SC coordinates, so that the user can map the location more accurately within the FOV. Note that in the case of a CB, the coordinates of the center are returned in SC coordinates; even if the center is not in the FOV. While the user is requested to supply normalized vectors, they are normalized again if possible; if not, then some vector is of zero length, an error message is returned for that invocation. It is necessary for the user also to supply one vector in or near the center of the FOV for two purposes:

a. to define the inside versus the outside[9]

b. to assist with certain other tests that will speed the calculation

It is user responsibility to ensure that the vectors $\mathbf{x}_i$ define the field of view well, i.e., no large gaps except on great circle arcs, vectors in order, no criss–crossing of the perimeter, etc., and to provide the vector near the center. That vector is used to perform a separate "conical hull" test described in Section 8.3, and in the Celestial Body case it is one of several tests for Earth blockage.

## 8.3 Conical Hull and a First Test Using it

Function:

- PGS_CSC_FOVconicalHull()

A test is developed here that simplifies many cases where the point clearly lies outside the FOV, and the same construction can also be used to determine if the Earth fills the whole FOV. The conical hull of the FOV is defined as the smallest circular cone centered on the i**nFOV** vector are containing the FOV. Refer to Figure 8–1. If the **nFOV** vector is optimally chosen, it will be the smallest circular cone containing the FOV. The algebra in Section 8.3 being fairly long, and that in Section 8.7 being worse, it is desirable to have a quick test to see if a point or an object might be in the FOV. In the case of a CB, the finite radius of the body slightly complicates the test, and

[9]This will avoid, for example, situations such as that in electromagnetic wave polarizations, wherein engineers define handedness from the standpoint of the receiver, and physicists from that of the source, so that "clockwise" means the opposite for the two. The methodology also avoids any ambiguity as to what is the inside and what is the outside of the FOV, in case it is large.

Figure 8–1 includes details for that case. Here it is only necessary to look at the inner circle defined by the heavier line. The radius rM of this circle, which is centered on the **inFOV** vector, is chosen so as to just include all the vertex vectors $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$, ....,.$\mathbf{x}_p$. To avoid unnecessary calls to trigonometric functions and their inverses, the tests are arranged as much as possible to use only the dot products of vectors. In this section, ignore the light, outer circle and the material referring to a "CB" (celestial body).

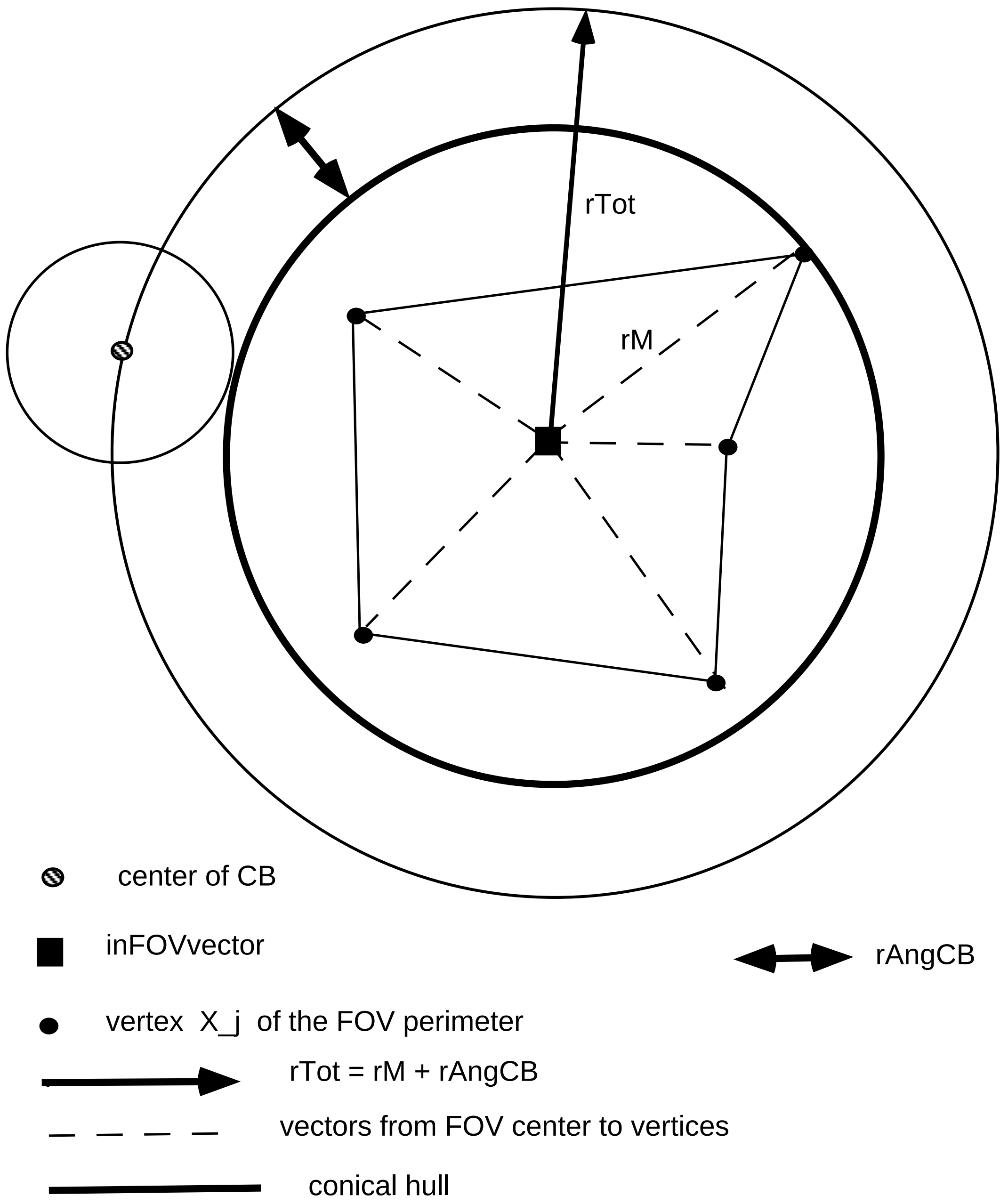

rTot
rM
center of CB
inFOVvector
rAngCB
vertex X_j of the FOV perimeter
rTot = rM + rAngCB
vectors from FOV center to vertices
conical hull

***Figure 8–1. Conical Hull Definition***

The method is simply to take the angle rM that is the arc cosine of the smallest dot product between the center vector and any other. In the present case, for the algorithm PGS_CSC_Earthpt_FOV(), *there is no need to find the angle rM; only its cosine, the smallest of the dot products*

$\text{dot}_i = \mathbf{inFOV}\ \mathbf{(*)}\ \mathbf{x}_i$ (for any i)

*is needed!* (Here **(*)** denotes the dot product.) Denote this number leastDot. Its arc cosine is rM.

(In the CB case, the value of the angle will be needed, because the CB angular radius must be added.) At this point, it is verified that leastDot > 0. If it is < 0, then the field of view is so wide that the algorithm may be unreliable, because it projects the vectors on the plane orthogonal to $\eta$, and an error message is sent back.

Now, it is easy to see that if the candidate point, $\eta$, is outside the circle of radius rM; it misses the field of view and needs no further work. That condition amounts to

$\boldsymbol{\eta}\ \mathbf{(*)}\ \mathbf{inFOV} < \text{leastDot} \Rightarrow \eta$ misses FOV

## 8.4 Solution of the "Point in FOV" Geometry Problem

Function:

- PGS_CSC_PointInFOVgeom()

### 8.4.1 Introduction

In the case of an Earth point it must be determined if a certain point defined in SC coordinates lies in the FOV, while in the case of a celestial body we start with a similar test on the center point of its disk. The common geometric problem is handled herein. Thus, this section solves the problem of finding whether a single point defined by a unit vector $\eta$ is in the FOV; the point is sometimes called the "candidate point."

The FOV perimeter and **inFOV** vector must be defined as in Section 8.2. Thus we assume that the FOV is defined by an ordered set of arbitrary perimeter vectors $\mathbf{x}_i$, i = 0,1,...p. (p = nFOV -1.) We will work the problem in SC coordinates, because the FOV has a complicated and variable shape in terms of its footprint on Earth. The Earth point or celestial body in question is mapped into SC coordinates using the Toolkit transformation before further processing. Figure 8–2 shows the basic geometry for the test.

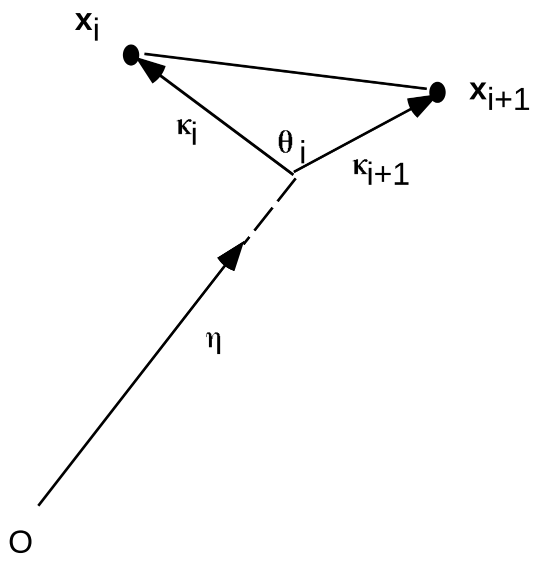


***Figure 8–2. How the $\kappa$ vectors swing around $\eta$***

### 8.4.2 Adding up the Angles $\theta_i$ Between Vectors $\kappa_i$ Swinging around $\eta$

To see if the point is within the FOV, from the tip of $\eta$, draw a line segment $\kappa_i$ to the tip of each vector $\mathbf{x}_i$. Let $\theta_i$ be the angle between $\kappa_i$ and $\kappa_{i+1}$, counted positive if counter–clockwise as viewed from the origin. Form totAng = $\Sigma_{\{i=0, \text{ to } i=p, \text{ and back to } i=0\}}\theta_i$. If $\eta$ lies in the FOV totAng will be $2\pi$, if outside, it will be zero, and if on the periphery it will be $\pi$. To allow for truncation, we count the point inside if the sum totAng exceeds 1.9 $*\pi$; otherwise outside. No attempt has been made to handle the case on the boundary (value $\pi$) because of truncation error and the possibility of inter–platform differences. If full spherical geometry was used in this analysis, the value would be $-2\pi$ for points outside, but we are assuming all the FOV perimeter vectors lie in a hemisphere about the inFOV vector and reducing the problem to the projection on a plane orthogonal to $\eta$. The effect is that totAng is zero when $\eta$ is outside the FOV unless it is in the "mirror" image of the FOV formed by a point reflection about the SC origin, when it will be $-2\pi$.

Construction of the vectors:

Let $\kappa_i = (\mathbf{x}_i - \eta\ (\mathbf{x}_i * \eta))/|\ \mathbf{x}_i - \eta\ (\mathbf{x}_i * \eta)\ |$

This is a unit vector orthogonal to $\eta$ and in the plane of $\eta$ and $\mathbf{x}_i$. The tip of this vector executes a closed curve (made up of line segments) in a plane orthogonal to $\eta$ as the index i runs from 0 to p and finally back to 0 again. We shall determine if the tip of $\eta$ lies within that curve by totaling the change in a polar angle measured counter–clockwise about $\eta$ as the index runs through its range.

Then we find the angle between the vectors $\kappa_i$ and $\kappa_{i+1}$ by the triple scalar product:

$$\theta_i = \arcsin (\kappa_i \mathbf{X} \kappa_{i+1} * \eta)$$

Here **X** is the vector cross product operator, and the triple scalar product with the dot and cross is used because is preserves the sign showing the sense in which the normal vectors $\kappa_i$ turn as i changes.

All the $\theta_i$ are summed and the test then applied. To obviate the problem that the user may have gone clockwise as seen looking out from the SC or looking in at the instrument; the test looks for the fiducial value obtained by using the **inFOV** vector in the center of the field as the first candidate vector, the true ones to be compared.

## 8.5 Earth Point in FOV

Tool:

- PGS_CSC_Earthpt_FOV()

This function will determine if an Earth point of given latitude and longitude is in the FOV. The test first transforms the point to SC coordinates by calling PGS_CSC_GEOtoECR(), PGS_CSC_ECRtoECI(), and PGS_CSC_ECItoSC(). It also finds distance Rs to the Earth center from the SC and determines if the candidate point is past the Earth limb, based on two approximate spherical Earth models—an inscribed sphere of radius B and a circumscribes one of radius A—this obviates further processing if the point is clearly behind the solid Earth. The test for being past the Earth limb (behind the solid Earth) is simply to use the Pythagorean theorem on D, the distance to the candidate point, and Rs, viz:

$$D^2 > Rs^2 - B^2 \qquad \text{condition I}$$

implies that the point is hidden, while

$$Rs^2 - B^2 > D^2 > Rs^2 - A^2 \qquad \text{condition II}$$

implies that the point would be hidden by the Earth if it filled the circumscribed sphere, so it may be behind the equatorial bulge. In the first case, the answer for point–in–FOV is reported as PGS_FALSE, while in the latter case the point is suspect, but it is impossible at this point accurately to determine if the point is behind the solid Earth. The method used is insensitive to Earth flattening and Earth aspect in the sense of the tilt of the Earth's axis in SC coordinates. If the test is marginal, (condition II), processing continues but a flag is set to check more accurately later, in case the point seems to be visible in the FOV. Since the final test in condition II requires a call to PGS_CSC_GetFOV_Pixel(), it is deferred until all other tests are passed.

The next test is conical hull test to see if the point is within a circular cone that envelops the FOV as tightly as possible. The point is reported as PGS_FALSE if not in the conical hull. This rapid test obviates the large cost of solving the actual geometric problem for an arbitrary shaped FOV (using PGS_CSC_PointInFOVgeom() ).

If the point is in the conical hull, processing continues. First, using PGS_CSC_PointInFOVgeom() (Section 8.4) to see if the vector to the point is in the FOV. If not, processing terminates. If so, then the point is reported in the FOV unless the flag was set previously by Condition II to indicate that it is closer than the Earth limb if the Earth is replaced by a circumscribed sphere, but farther than if the Earth was replaced by an inscribed sphere. In that case, PGS_CSC_GetFOV_Pixel() is called to determine, with full account of Earth figure and aspect, if the point is closer or farther than the limb. If the latitude and longitude returned by PGS_CSC_GetFOV_Pixel() agree within 0.0001 of the original, the point is deemed to be visible; else it is behind the bulge of the Earth and not visible. The number 0.0001 is picked as a compromise between machine accuracy and the effects of terrain and atmosphere. Too large a tolerance would lead to a false report that a point behind the Earth is in the FOV (in this rare case where Earth oblateness is important). Too small a tolerance could lead to a false negative result due to roundoff, and the effects of terrain and refraction.

## 8.6 Celestial Body in FOV

Tool:

PGS_CBP_body_inFOV()

This tool determines if any portion of a Celestial Body "CB" is in the FOV (field of view). A celestial object cannot be in the FOV if the Earth intervenes. Therefore the tool checks for this condition and reports PGS_FALSE in that case. The Earth is not considered a celestial body for the purposes of this tool. There is a separate tool to determine if an Earth point is in the FOV.

### 8.6.1 Introduction:

The user supplies a celestial body identifier (cbID) (see Appendix G), a time or array of times, and an FOV specification. The function accesses the CBP DE200 sun/moon/Planetary Body ephemeris via the tool PGS_CBP__Sat_CB_Vector() to locate the body in spacecraft coordinates. If the user wishes to know if a star or user defined point is in the FOV, the identifier PGSd_STAR can be supplied in place of one of the standard celestial body ID's. In that case, the ephemeris tool PGS_CBP_Sat_CB_Vector() cannot be used for the coordinates, which must be supplied by the user. The coordinates are transformed automatically from ECI to SC coordinates within PGS_CBP_body_inFOV() by a call to PGS_CSC_ECItoSC(). The FOV is to be supplied by the user as a sequence of vectors in spacecraft (SC) coordinates bounding it (Section 8.1). It is important, from the user's standpoint, that a celestial body has finite size. The reason is that even a small part of the moon or a bright planet can introduce significant light in the FOV. It is therefore inadequate to test only if the body's center is in the FOV. An exception is stars (other than the sun), whose angular radius is, for practical purposes, zero.

The CB has angular radius rAngCB, determined as follows: let rCB be the distance to the body, and rLinCB be its linear radius. (rLinCB is fixed for a given invocation of this function, by Table E–1 in Appendix E, but rCB varies with the distance.) Then

$$\text{rAngCB} = \arcsin(\text{rLinCB}/\text{rCB}) \tag{8.6–1}$$

In this function, to save computation, rAngCB is calculated only for the first time in an invocation, except in the case of the moon, whose distance changes rapidly. It has been necessary to make some compromises in defining the sizes of the planets, because several of them have notable satellite and ring systems. In Appendix G, these compromises are listed, along with the actual assumed linear dimension. The PGS CB Ephemeris tool PGS_CBP_Sat_CB_Vector() provides the locations of all the celestial bodies in the Table, in SC coordinates, corrected for aberration and parallax. The sizes in the Table are converted to meters internally before use in the present function, because it is the angular radius that matters, and this is, in radians, approximately the radius divided by the distance.

This tool is quite complicated, because of the details of possible Earth blockage. In some cases, the results PGS_FALSE can be reported early on for CB–in–FOV, because the CB is outside the conical hull, or the Earth blocks the whole conical hull, or the Earth occults the entire CB. When none of these conditions hold, it is possible that the CB lie all or partly in the FOV, yet the Earth occults the portion of the CB that would otherwise lie in the FOV. The test for the Earth to block just that part of the CB is fairly complicated, so the test for this type of partial blockage is deferred as late as possible.

The tool works entirely in SC coordinates. On output, the algorithm provides to the user the actual vector to the CB in SC coordinates, to permit hand checking, subdivision of the FOV, or other tasks of user interest. This answer is provided whether or not the object is in the FOV. The conical hull test and most of the Earth blockage tests are done early on, and will be described first. If all else indicates that the body is in the FOV, but the part of it in the FOV might be occulted by the Earth's equatorial bulge, then a more sophisticated test is used at the very end.

### 8.6.2 Conical Hull Test—Test for a Celestial Body

Function:

- PGS_CSC_FOVconicalHull()

The cost of later tests for overlap of the CB with the FOV, using the detailed description of its periphery, is fairly high, so the conical hull test is performed first on each vector before passing it to the lower level function. The call to the lower level geometric function PGS_CSC_PointInFOVgeom() will pass in a PGS_FALSE for any $\eta$ in the array the case of missing the conical hull. *The case of a vector $\eta$ missing the conical hull is tested first and all the operations in PGS_CSC_PointInFOVgeom() are skipped in that case, although in the case of processing many time offsets at once, the function may be called.*

In ascertaining if the CB disk lies outside the conical hull, the situation is very much like the one for a single point, such as an Earth point, but one essentially pads the hull with a ring whose width is the radius of the CB. Please refer again to Figure 8.1 for the geometry of the conical hull test for this case. Now, a “guard circle” of larger radius rTot must be placed around the ordinary conical hull, to define a region where the CB center might lie and the CB disk overlap the hull. The calculation runs:

rM = arc cos(leastDot) (8.6–2)

rTot = rM + rAngCB (8.6–3)

Again, to minimize the use of relatively slow trigonometric functions, the algorithm tests against with dot products, viz:

grossDot = cos(rTot) (8.6–4)

followed by the test

$\eta$ **(*) inFOV <** grossDot => CB entirely outside FOV (8.6–5)

instead of testing against leastDot. (Note that grossDot, despite its name, is less than leastDot, because the cosine and arc cosine are decreasing functions of their arguments.) Again, PGS_TRUE is passed with the array member candidate vector if the foregoing condition does not hold, PGS_FALSE if it does, and furthermore, again, the *case of all vectors missing must be tested first and the entire call to PGS_CSC_PointInFOVgeom() skipped in that case!*

### 8.6.3 Earth Blockage: Method

Function:

- PGS_CSC_EarthOccult()

This test comprises a separate function that determines if the celestial body (CB) could be in the field of view (FOV). The test will bypass the entire CB test when the CB lies entirely behind the Earth, *or* when the Earth is known to fill the entire field of view. The test is in three phases; the last of which is implemented in PGS_CBP_body_inFOV(); using the vector “extremeVec” returned by PGS_CSC_EarthOccult(). The first phase does not depend on the CB at all—it is just a check if the Earth fills the field of view. It and the second test are based on the overlap of round images: the FOV conical hull or the CB image, with a round image whose radius on the sky is the Earth's semi minor axis. Thus, these tests will miss a few cases where the CB is blocked by the Earth's equatorial bulge. The first test is just a screening test. The second test is nearly definitive. This second test (exercised only if the first fails to find total occultation) will require the entire CB to lie within the circle inscribed in the Earth. If this test fails by an amount comparable to the difference in radius (~ 13.5 mi or 21.5 km) of the Earth at equator and pole, the more sophisticated third test will be applied later. The first test depends on the FOV Conical Hull size and center vector but not on the CB radius nor location. Therefore, when it detects blockage, the tool PGS_CBP_SCtoCB_Vector() need not be called. The second and third tests depend on the CB radius but not on the FOV at all. None of the tests depends on the FOV shape within the Conical Hull. The first two tests, therefore, are run as a screen to reduce the calculational burden of determining if the CB overlaps the actual FOV; also, however, the three tests constitute the only check for Earth occultation. The third test is run only after the underlying tool has determined that the CB overlaps the FOV, because it is not perfectly reliable.

All the tests require that one know the vector to the Earth center in SC coordinates, which will be denoted **scToEcenter**. The norm of this vector is

scToEC = norm(**scToEcenter**) (8.6–6)

The unit vector from SC to Earth is

unitEarthVec = scToEcenter/scToEC (8.6–7)

### 8.6.3.1 Test 1: Total FOV Blockage

A disk on the sky inscribed in the oblate Earth has an angular radius on the sky given by

angRadE = arcsine(C/scToEC) (8.6–8)

where C is the semi–minor radius of Earth and scToEC the distance to Earth center. For a later use in Test 3, let us also define

bigAngRadE = arcsine(A/scToEC) (8.6–9)

The test will use the half angle rMax of the enveloping cone or conical hull of the FOV. As was just explained, the cone will generally be smaller than the Earth on the sky. That is the only case where one can apply this test. In this case, one tests if the angular distance from the **inFOV** vector to the Earth center is less than the angular Earth radius angRadE minus the cone half angle rMax. The test goes as follows:

Let

EarthDotInFOV= **unitEarthVec** (*) **inFOV** (8.6–10)

then

arccosine(EarthDotInFOV) < angRadE - rMax => FOV occulted by Earth (8.6–11)

But

angRadE = arcsine(C /scToEC) (8.6–12)

and we can use identities to eliminate angles. Refer to the identities 4.4.33 and 4.4.35 on p. 80 of the U.S. Department of Commerce *Handbook of Mathematical Functions.* (U.S. Government Printing Office, 1954). It is desirable as well to express rMax as

rMax = arccosine(rMdotProduct) (8.6–13)

Then also identity Equation (4.4.33) can be used to combine rMax with the left hand side of Equation (8.6–11), after which Equation (4.4.35) is used to combine with the angRadE. Thus no trig functions will be used. The algebra looks simpler if one uses abbreviated names.

Let:

E stand for EarthDotInFOV (8.6–14)

r stand for rMdotProduct (8.6–15)

and

p stand for C/scToEC (8.6–16)

Then if

q = E*r - sqrt((1-E*E)*(1-r*r)) (8.6–17)

the Earth occults the FOV if and only if

p*q > sqrt((1-p*p)(1-q*q)) (8.6–18)

where the cited Equations (4.4.33) and (4.4.35) were used in that order.

#### 8.6.3.2 Test 2: CB Behind Earth

The CB has angular radius rAngCB. If the angle ang_CB_Earth from CB center to Earth center plus the radius rAngCB is less than angRadE, then the whole CB disk is occulted by Earth and it is not visible. Then there is no need to check if the CB is in the FOV.

Let

$\eta$ = scToCBvec /norm(scToCBvec ) (8.6–19)

scCBdotSCearth = $\eta$ (*) unitEarthVec (8.6–20)

The angle angCB_Earth may be found from

angCB_Earth = arccosine (scCBdotSCearth) (8.6–21)

This test does not involve the size or shape of the FOV. The CB will be occulted if

angRadE > angCB_Earth + rAngCB => CB occulted by Earth (8.6–22)

which amounts to

arcsine(C /scToEC) > arccosine (scCBdotSCearth) + rAngCB (8.6–23)

Again use the identity 4.4.35 on p.80 of the U.S. Department of Commerce *Handbook of Mathematical Functions*. As before, let p stand for C /scToEC, and also let

s stand for scCBdotSCearth (8.6–24)

then Equation (8.6–23) becomes

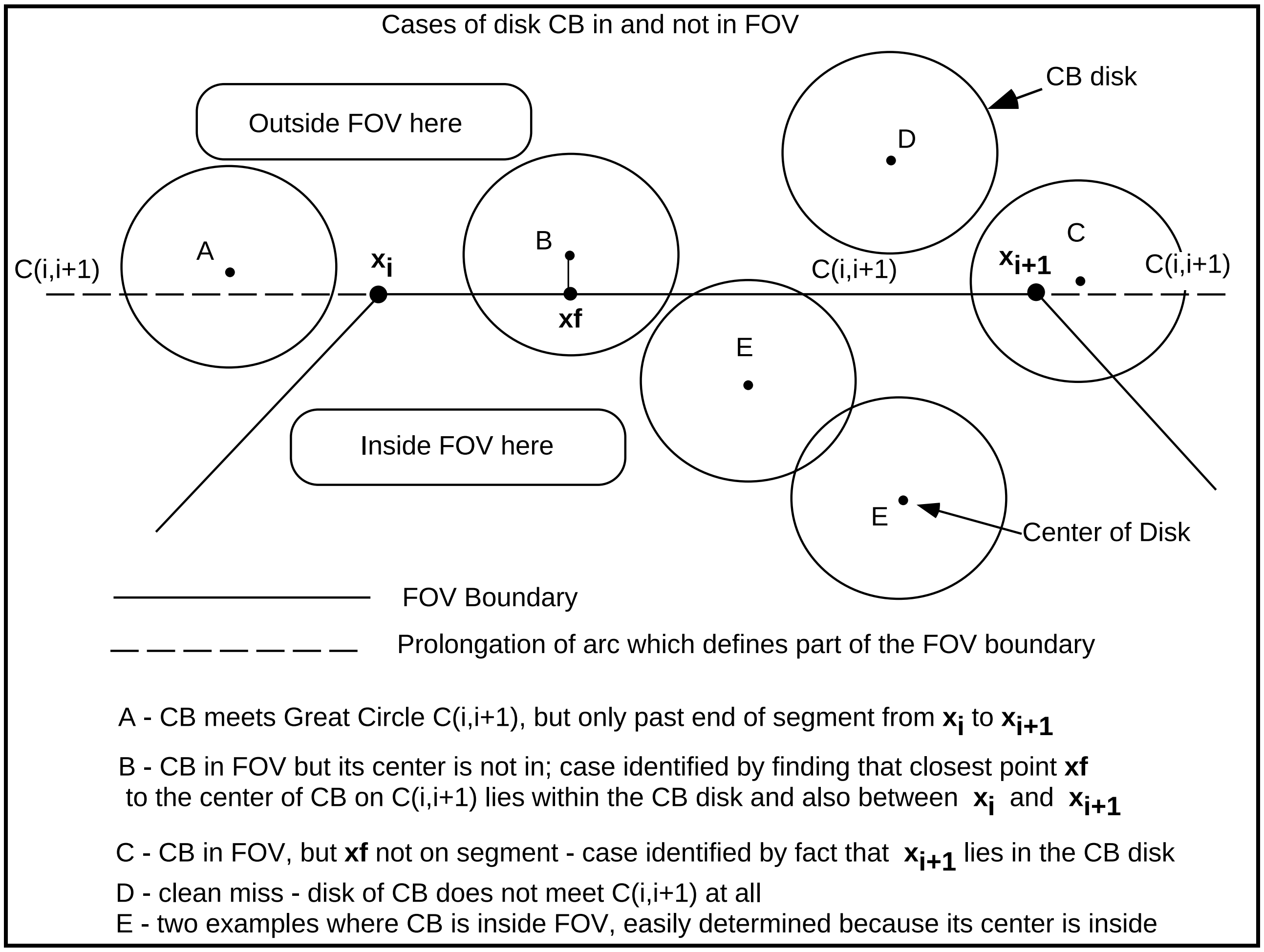
p*s - sqrt((1-p*p)(1-s*s) > sin(rAngCB) => CB occulted by Earth (8.6–25)

In test phase Equation (8.6–25) was checked against Equation (8.6–24)

If condition (8.6–25) holds, the CB is hidden by the Earth. If it does NOT hold, the test is repeated with the replacement angRadE -> bigAngRadE, and the result saved. Let us call it bigBlock = PGS_TRUE if true; that means that a disk of angular radius bigAngRadE on the sky, centered on Earth center, would occult the CB. If neither Test 1 nor Test 2 indicates blockage, the basic FOV test is applied, but if bigBlock is PGS_TRUE then a final test is applied later if the CB appears to overlap the FOV. To keep the discussion of the blockage tests together, the last test will be explained now, however.

### 8.6.4 Finding if the CB Disk Overlaps the FOV—General Case

In this section, it is assumed that the CB has passed the simple tests on lying entirely outside the conical hull or behind the Earth, and that the Earth does not block the entire FOV. Thus, one needs to know if the CB disk overlaps the FOV. Imagine that the instrument is at the center of a unit sphere or the celestial sphere, on which is traced out a field of view (FOV), and on which the CB lies. The geometric problem in this algorithm is to determine if the circular cap (herein called a disk) that represents the CB on the celestial sphere intersects the FOV, defined by a set of vectors around its periphery. There are numerous cases, illustrated in Figure 8–3. The cases will be discussed with parenthetic reference to that figure. If the CB center is inside the FOV (E), the solution is immediate. Otherwise, the CB may lie totally outside (A,D), or intersect the FOV perimeter along a bounding segment (B), or at a vertex (C). The problem is solved by considering all these cases, the easiest to solve first.



A - CB meets Great Circle C(i,i+1), but only past end of segment from $\mathbf{x_i}$ to $\mathbf{x_{i+1}}$

B - CB in FOV but its center is not in; case identified by finding that closest point **xf** to the center of CB on C(i,i+1) lies within the CB disk and also between $\mathbf{x_i}$ and $\mathbf{x_{i+1}}$

C - CB in FOV, but **xf** not on segment - case identified by fact that $\mathbf{x_{i+1}}$ lies in the CB disk

D - clean miss - disk of CB does not meet C(i,i+1) at all

E - two examples where CB is inside FOV, easily determined because its center is inside

***Figure 8–3. Possible Configurations of the CB and FOV Periphery***

The function uses a subfunction PGS_CSC_PointInFOVgeom() which determines if a point lies within the FOV. The unit vector to the point in question in SC coordinates is denoted $\eta$. Given an FOV whose perimeter is defined by nFOV unit vectors $\mathbf{x}_i$, i = 0,1,...p (listed in order so as to complete an almost closed curve), the function PGS_CSC_PointInFOVgeom() determines if a unit vector $\eta$ lies inside or outside the FOV. In the present case, $\eta$ will represent the center of the celestial body.

Our problem is to determine if any part of the disk diskCB, of angular radius rAngCB, centered at $\eta$, lies inside the FOV. rAngCB is defined in Equation (8.6–1). The work proceeds as follows, in outline:

a. Determine if $\eta$ lies in the FOV. If so, then obviously part or all the CB disk lies in the FOV, and the tool reports PGS_TRUE. This is case E of Figure 8.3

b. If $\eta$ does not lie in the FOV, then one must determine if the disk intersects any of the boundary curves $B_i$, which run from $\mathbf{x}_i$ to $\mathbf{x}_{i+1}$. [The last $B_i$ curve is from $\mathbf{x}_p$ to $\mathbf{x}_0$.] If any intersection exists, the answer again is PGS_TRUE, else PGS_FALSE. It is not difficult to determine if the diskCB intersects the great circle C(i,i+1) through $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, but if so, it is necessary to ensure that the intersection (or part of it) is on the arc from $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ and not entirely on the prolongation of that arc. This problem will be solved in due course. Let the case that the disk intersects neither the arc nor its prolongation be denoted the case of a "clean miss." This is case D of Figure 8–3. [The C(i,i+1) list is to be numbered so that "i" wraps—in other words, C(p,nFOV) really means C(p,0).]

The algorithm must proceed arc by arc in this phase. There is a certain setup cost that will become apparent. It decided for efficiency first check all the candidate points $\eta$ for inclusion in the FOV. This is done with PGS_CSC_PointInFOVgeom(). If the CB center point is in the FOV, then obviously so is part of the CB. If some of the $\eta$ vectors (for some of the time offsets) lie in the conical hull but not in the FOV, more work is needed. Certain scratch vectors **cp** calculated later depend only on the FOV, not on $\eta$. Thus, the setup is done before the test for disk intersection before the different $\eta$ vectors are run through. The test must be performed arc by arc, but it is finished if any PGS_TRUE value is obtained and the arc–by–arc loop can then be exited. This applies at each stage of the search: so long as inFOVflag is PGS_FALSE the search must be continued, but if true, then it is appropriate to proceed to the next time offset.

If the CB is not a "hit" or "clean miss," it becomes necessary to check the condition (for each i) that $\mathbf{x}_i$ lie in diskCB. This amounts to the condition $\mathbf{x}_i$ (*) $\eta$ > cos(rAngCB), where (*) means dot product. It is best to check this condition first, before examining mid–arc intersections. If it is true, one of the vertices of the FOV lies within the diskCB [Case (C)], so perforce part of the diskCB is in the FOV; the answer can be reported as "PGS_TRUE" and one passes to the next candidate $\eta$ (assuming an array of times has been supplied). This test should be done first, vertex by vertex, because if the test is deferred until an arc by arc search, it needs to be performed at each end of the relevant arc. Also, it may turn out that there is a definite answer on each $\eta$ vector without setting up the **cp** vectors, which are needed only in the case where both of the following pertain:

(I) not a "hit" ( hit means $\eta$ inside FOV) and

(II) no $\mathbf{x}_i$ vector lies in disk CB

Assuming that there is some $\eta$ that fails one of conditions (I) and (II), then an arc by arc analysis as in the next section is required to see if the CB disk has one of the bounding arcs as a chord. In this regard, note that it is impossible for the CB disk to meet a bounding arc ($\mathbf{x}_i$ ,$\mathbf{x}_{i+1}$) unless either it overlaps an end of the arc (condition II), or else a perpendicular from its center to the great circle defined by ($\mathbf{x}_i$ ,$\mathbf{x}_{i+1}$) meets the circle between the two points, i.e., on the arc. The proof is left to the reader.

### 8.6.5 Further Details of the Arc by Arc Analysis

#### 8.6.5.1 Determining if CB Disk Intersects a Boundary Arc or its Prolongation

Here we determine if "clean_miss” is PGS_FALSE

The cross product **cp** = ( $\mathbf{x}_i$) **X** ($\mathbf{x}_i$) / | ( $\mathbf{x}_i$) **X** ($\mathbf{x}_i$) | is the pole of the great circle C(i,i+1) through $\mathbf{x}_i$ and $\mathbf{x}_i$. It is easily seen that there is a clean miss if η is within the arc distance (π/2 - rAngCB) of either **cp** or **-cp** (the anti–pole). Consider the quantity

| acos(**cp** (*) η**) - π/2** | < rAngCB,

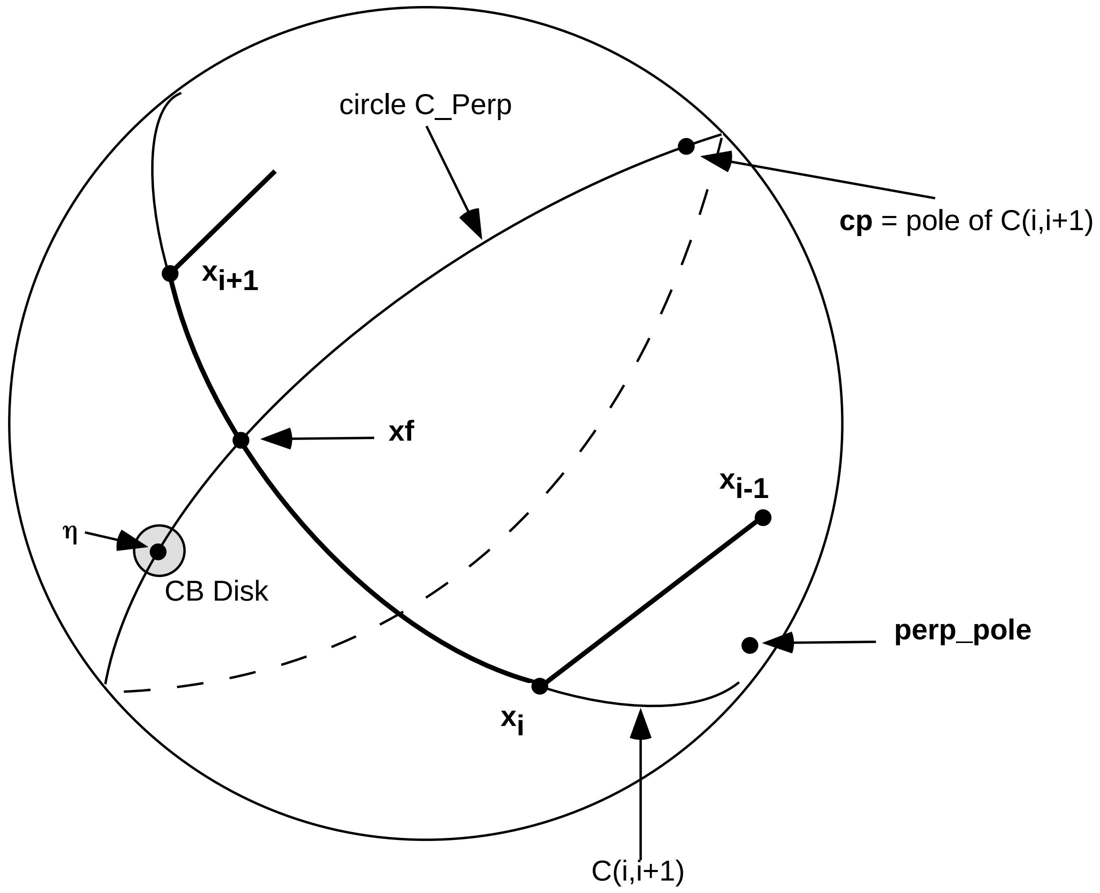
where the bars "| |" indicate absolute values. If this condition holds, then clean_miss = PGS_TRUE. Otherwise, it is PGS_FALSE.

### 8.6.5.2 Determining if CB Disk/FOV–boundary Arc Intersection is on the Arc Itself

If the disk intersects the arc or its prolongation into a great circle, it is necessary to determine whether any of the disk meets the arc between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ [Cases (B) or (C)]; otherwise it meets only the prolongation [Case (A)]. The simplest test is to see if either end on the arc lies in the diskCB. This test amounts to the condition **x_**k (*) η > cos(rAngCB), where (*) means dot product and k stands for either i or (i+1). It can now be seen that if the tests are done arc by arc, then the vertices will need be tested two at a time (those at the ends of an arc whose great circle intersects the diskCB). The test was done first, vertex by vertex, which excludes Case (C), so need not be done again. (See Section 8.6.4)

For the geometrical description see Figure 8.4

Finding **xf**, the closest point to CB on one of the great circles C(i,i+1) bounding the FOV by constructing the orthogonal circle C_Perp, defined by its pole **perp_pole**.



Observer is at center of the sphere. Heavy arcs are part of the FOV perimeter. The FOV itself is towards the upper right. In this example, CB is outside FOV. Small solid points, such as **perp_pole** and η are all the tips of vectors from the origin to the sphere's surface, which represents the sky as seen by the observer.

***Figure 8–4. Great Circle on the FOV Boundary and an Orthogonal Great Circle Through the CB***

If neither end of the arc lies in the diskCB, then either the arc segment ($\mathbf{x}_i$, $\mathbf{x}_{i+1}$) is a chord of diskCB, or it lies entirely outside it Cases (B) or (A). To check these cases one must find the point nearest to the CB center on the great circle coincident with each bounding arc. Thus, one determines if the diskCB meets the great circle C(i,i+1) or outside that segment by determining if the foot of the shortest perpendicular from $\boldsymbol{\eta}$ to the great circle through $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ lies between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. The vector to the foot of the perpendicular will be denoted **xf**.

The perpendicular from $\eta$ to C(i,i+1) is a great circle whose pole is perpendicular to both $\eta$ and the pole of C(i,i+1) (in other words, it lies on C(i,i+1). Using cross products again, we find that the pole of this perpendicular circle C_Perp is

**perp_pole = cp X $\eta$ / | cp X $\eta$ |**

where **cp** = $\mathbf{x}_i$ **X** $\mathbf{x}_{i+1}$ / | $\mathbf{x}_i$ **X** $\mathbf{x}_{i+1}$ |

(Note: if **cp X $\eta$** = 0, then **perp_pole** is undefined, but in this case, the CB is 90 degrees from the arc, and it must be either in the FOV by the earlier test, or must be a "clean miss." A Toolkit error issues if a zero cross product is detected at the present stage). The two intersections of the great circles C(i,i+1) and C_Perp will be two points **xf** = ($xf_1$,$yf_1$,$zf_1$) and ($xf_2$,$yf_2$,$zf_2$) which are simultaneous solutions of

**xf (*) cp** = 0

and

**xf (*) perp_pole** = 0

These are only two equations in three unknowns, but we may assume a unit vector (actually a different solution method is used—see below). Note: If the two 3–vectors of coefficients are parallel, there is no solution, but that means that $\eta$ lies on C(i,i+1). This condition is unlikely, because the test for the object's center being in the FOV using PGS_CSC_PointInFOVgeom() is expected to report positive in this case. Yet it possible, in view of the limited testing, that PGS_CSC_PointInFOVgeom() might fail for a point on the actual periphery. In that case, $\eta$ lies on C(i,i+1), and the present test ought to report a positive answer, because the CB will surely intrude into the FOV (except in the case of a star, of zero apparent radius, but we can report positive in that case with no real problem.) Therefore, if the denominator is zero, the present function will set the answer to PGS_TRUE for CB in FOV.

Instead of solving the two simultaneous equations, it was decided to use the normalized cross product:

**xf = cp X perp_pole / | cp X perp_pole |**

and its negative as the two vectors that could represent the point on C(i,i+1) closest to the CB. If the cross product is zero, the two vectors are parallel or anti–parallel, which is the singular condition just discussed; in that event, we set the flag for CB in FOV to PGS_TRUE and terminate processing for this time offset. Of the two vectors $(xfoot,yfoot,zfoot)_1$ and $(xfoot,yfoot,zfoot)_2$ (which are diametrically opposite, so that one need solve only for one) we must choose the one nearest $\eta$, here denoted **xfNear**. The one near the CB can be found by testing **$\eta$ (*) xf**; the positive dot product indicates the right one. It will lie on C(i,i+1) and we must determine if it lies between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$. The simplest test is to take the dot product of vectors from **xfNear** to $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$, which will be negative if it is in between and positive otherwise. If negative, we have the case inFOVflag = PGS_TRUE and we are done with this time offset. Incidentally, it is impossible for **xf** to lie on the prolongation of the arc C(i,i+1), but not on the arc, and for the CB disk to overlap the FOV, and yet for the CB disk not to contain either $\mathbf{x}_i$ or $\mathbf{x}_{i+1}$. In other words, Cases (A)–(E) are exhaustive. The interested reader may wish to convince her or himself with diagrams or geometric proofs.

### 8.6.6 Test 3: Does Furthest Point Of CB Disk Miss Earth Limb?

Test 3 is for Earth blockage undertaken only if the CB is found to lie in the FOV and bigBlock = PGS_TRUE. In that case, the CB overlaps the field of view, but would be occulted by the Earth if the Earth were a sphere of radius A—the semi–major axis. The idea of the test is that the most likely point on the CB disk to peep past the Earth limb is the point farthest from Earth center; we test if that point is not obscured, and is in the FOV. The test is not absolutely definitive, but will discard a few cases where the Earth's bulge (difference in radius over that of an inscribed sphere) occults the CB. The method is to construct a vector in SC coordinates that points at the part of the CB most distant from Earth center, which is called the “extremepoint,” and then to apply PGS_CSC_GetFOV_Pixel to see if that line of sight intersects Earth. If so, the body is occulted. If not, it is still possible that it misses the FOV. The probability is very small, because the Earth is nearly spherical and it is unlikely that the FOV overlaps the CB, but is so arranged that it does so only on the portion hidden by Earth. It would be possible to test if the extreme point lies in the FOV by applying PGS_CSC_PointInFOVgeom() to the extreme point, but the situation seems so improbable that this was not done.

The determination of the extreme point is shown in Figure 8–5. This figure should be interpreted cautiously, because it is a plane figure representing plane geometry, but there are also arcs joining points E, B, and Q on the unit sphere. Plane geometry in the common plane of the vectors is perfectly valid, of course.

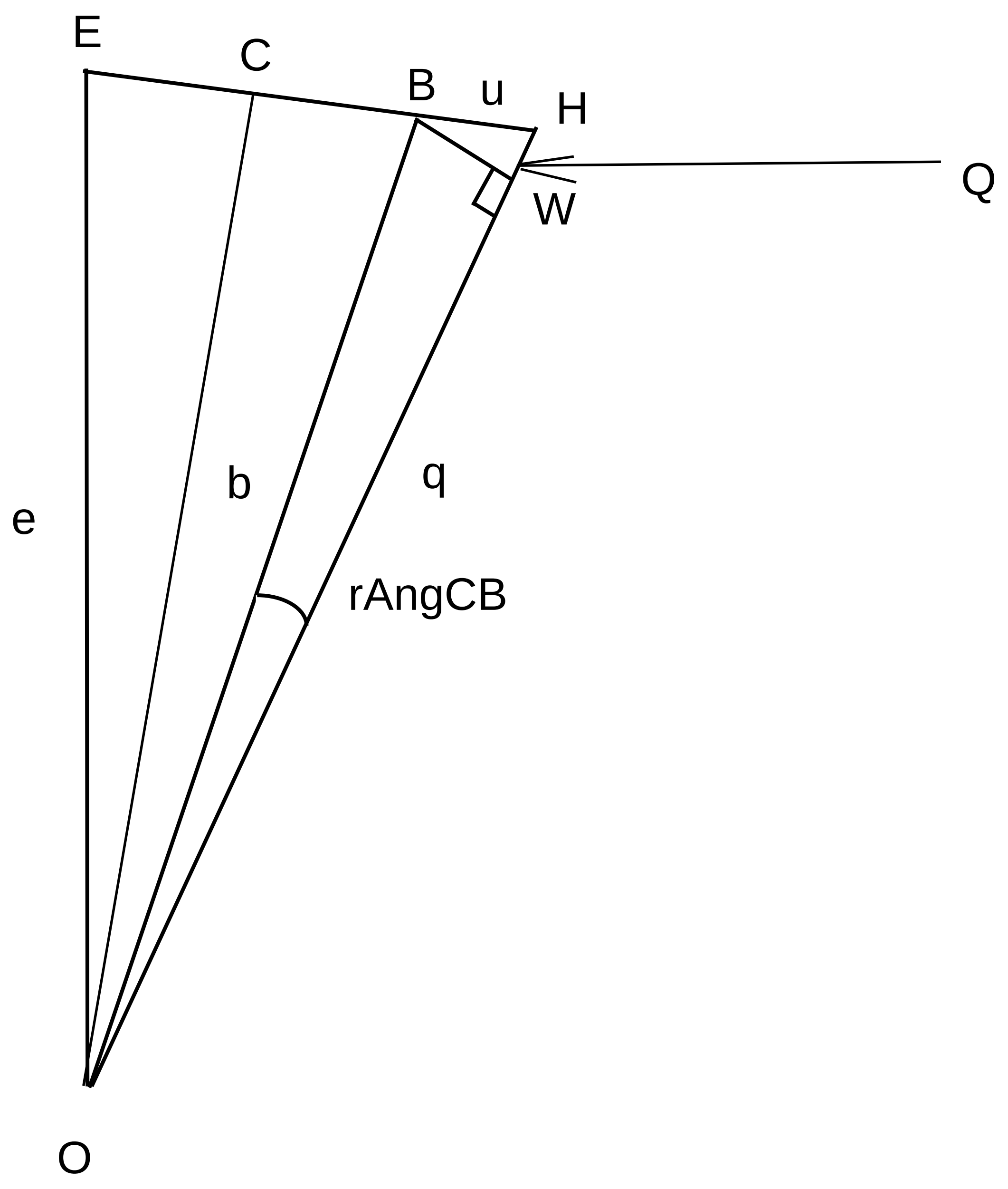


***Figure 8–5. Point of the CB Most Extreme from Earth Center***

The vector extremeVec is constructed by rotating the vector to the CB by rAngCB, away from Earth center. For brevity denote the spacecraft as O (the origin of coordinates), the image of the Earth center on the unit sphere about O as E, and the center of the celestial body as B. Denote a straight line between two points by concatenating the two letters as in EB representing the line from E to B. The boldface version will represent a vector along the line. Let the line of **q** (i.e., **q**, prolonged) intersect the line EB at H; both Q and H will then lie on **q** somewhat beyond the tip of **q**. Q will be closer to O because **q** = **OQ** is a unit vector. Two methods will be given for finding q. The first involves the straightforward solution of simultaneous equations. The second is more geometrically motivated and shorter. The idea in the second method is that it is much easier to find **OH** than **q**, and we can afterwards normalize **OH** to get **q**. The coordinates of the point H are obtained from those of B by constructing the vector **BH**, which is clearly a multiple of **EB**.

For the first method denote

**q** = extremeVector = **OQ**

**b** = scToCBunit = **OB** ( = $\eta$ )

e = unitEarthVec = OE

Then clearly, the following vector equations hold:

**q** = a **b**+ d **e**, where a and d are constants

(This holds because the vectors are coplanar and **b** and **e** are linearly independent.)

**q** (*) **e** = cos(rAngCB)

| **q** | = 1

These are three equations for the three components of **q**, which can then be found. It is used in PGS_CSC_GetFOV_Pixel as described, completing the last test.

Solving the foregoing equations proved to be somewhat lengthy. Therefore a shorter method was devised—a method that also avoids the ambiguity inherent in solving a quadratic system. Let

**uEB** = **EB** / | **EB** |

where **EB** = **b - e**

Then once we find the distance u = | **BH** | we can find H by

**OH = b + u * uEB**

To find u, drop a perpendicular from **B** to **OH**, meeting it at point W. Then because **b** is a unit vector,

| **BW** | = sin(rAngCB).

Denote an angle by vertices in order, so that, for example, the angle between **e** and **b** is EOB. The distance u can be found most easily by constructing the bisector OC of the angle EOB; meeting EB at C. Then angle OCB is a right angle. The triangles OCH and HBW are similar, because side OC is perpendicular to side BH and side OH is perpendicular to side BW. Therefore, angle

HBW = COH = rAngCB + COB = rAngCB + (EOB/2)

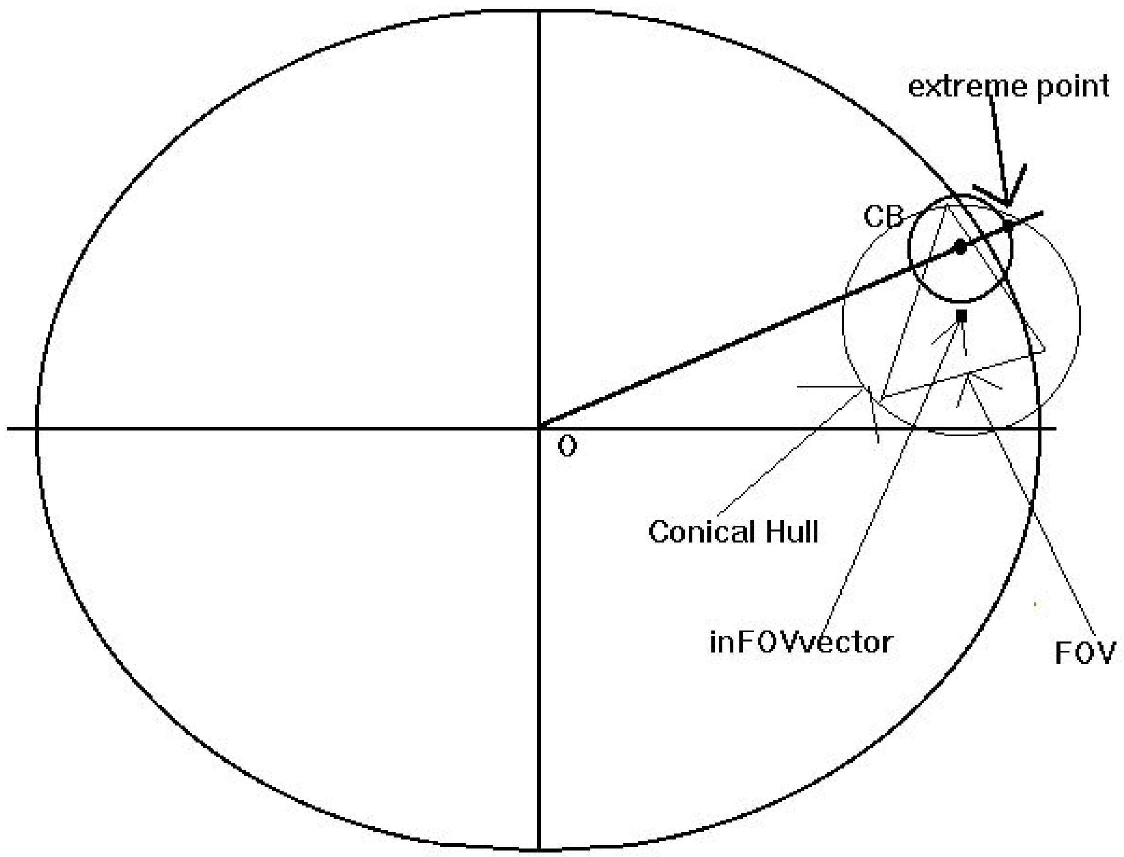
Finally, then,

u = BW sec(HBW) = sin(rAngCB) sec(rAngCB + (EOB/2)),

where, of course,

EOB = acos (**b** (*) **e**).

If these tests are passed, then the CB may lie in the FOV and detailed analysis is needed, as follows. Before proceeding, we give one caution: The foregoing Earth Occultation test is not quite definitive. It is conceivable that the Earth disk, the FOV shape, and the CB location conspire so that the CB overlaps the FOVonly in a region behind the Earth limb, while both the body and the FOV protrude past the Earth limb, but in regions that do not overlap. In this case, our software would report the body in the FOV when it is not. The geometry for this exceptional case is illustrated in Figure 8–6.



In this stressful example, the CB disk is not totally occulted by the Earth, and falls partly in the FOV (triangle), but the portion of the CB that protrudes past Earth limb fails to intersect the FOV. Therefore, the object could falsely be reported as in the FOV when it is not. This will be extremely rare.

***Figure 8–6. Exceptional Case***

This is expected to be an extremely unlikely event and we have judged in our designing the software that the user would prefer a “false positive” to a “false negative”.

### 8.6.7 Additional Remarks And Cautions; Earth Model Issues

Some tests are also performed on input vectors:

a. No input FOV input vector may be zero length; if non–zero, it is re–normalized to unit length. Warning to user is

   PGS_CSC_ZERO_FOV_VECTOR

b. No two consecutive input FOV vectors should be collinear nor diametrically opposite (in latter case there is no way to define which way you proceed from one to the other). Warning to user is

   PGS_CSC_COLLINEAR_FOV_VECTORS

The actual check of Earth occultation is performed using the WGS84 model of the Earth, with no atmosphere. This method is chosen to avoid having to access an Earth model data base. Most cases can be disposed of quickly without any allowance for precession or nutation, but if the line of sight to the CB passes close to the Earth limb the tool invokes PGS_CSC_GetFOV_Pixel() to provide a very accurate check for the line of sight intersecting the Earth. The calculation includes precession and nutation (which affect the projection of the Earth spheroid on the sky) and aberration for the SC velocity; however, the geoid and terrain are not considered. Obviously, limb sounders may be concerned with more accurate determination of the exact relationship of a vector to a CB with the terrain and the atmosphere. This question can be addressed by using various transformations in the toolkit. It is also possible to use the present tool, but to use PGS_CSC_GetFOV_Pixel() to seek Earth intersection. In that case the user can supply her/his own chosen Earth model or use a toolkit–supplied model. For example, a somewhat inflated Earth model would yield a latitude, longitude, and altitude relative to that model, for a vector that narrowly missed the WGS84 model. The Toolkit models are in the file /lib/database/CSC/earthfigure.dat, that can be edited by the user to add models of her/his choice; however, only spheroidal models (one major, one minor axis) are accommodate d. (We ignore diffraction and glints; it is user responsibility to consider glints, for example by adding additional FOV definitions.) The tool does not check for eclipse of one celestial body by another, but only for occultation by the Earth. The only case of eclipse that is significantly probably is by the sun or the moon; thus it is advised to check these objects first; if one of them is in the FOV, it will dominate.

This page intentionally left blank.

# 9. Economies and Shortcuts

The following shortcuts and economies were taken to speed the calculations with no significant degradation of accuracy.

## 9.1 Ephemeris (EPH)Tools

Position interpolation done with a cubic fit to nearest two points only (this is under further study)

Attitude and attitude rate interpolation are done independently and only kinematically, not dynamically

## 9.2 Time and Data (TD) Tools

TAI as seconds from 1 Jan. 1993 are carried as only one double precision, not two.

## 9.3 Celestial Body Position (CBP) Tools

a. The sum of stellar aberration and planetary light travel time is replaced with planetary aberration.

   accuracy: 0.1" arc

b. Terms of order $(v/c)^2$ in aberration are ignored.

c. Aberration due to spacecraft motion is simply added to the planetary term.

d. In seeking Celestial Body (CB) in FOV, the angular size is calculated from the physical size and distance using the first time value in the call, unless the object is the moon. This neglects the effect of changing distance except for the moon, which is quite accurate over periods of hours. In the case of the moon, the apparent diameter can change rapidly due to spacecraft motion.

e. In seeking Celestial Body (CB) in FOV, an approximation is made in one difficult case. In this case where the CB is in the part of the FOV blocked by the Earth’s equatorial bulge, but not in the part of the FOV that is not blocked, the result may be reported as in the FOV, which is incorrect. The case will be extremely rare.

## 9.4 Coordinate System Conversion (CSC) Tools

In PGS_CSC_GetFOV_Pixel, the low accuracy case omits several calculations, including Earth rotation during light travel.

Refraction is ignored (user can put a refraction correction in by using Zenith Angle)

Topography is ignored

TDT–TDB correction uses a simple equation from Astronomical Almanac

## 9.5 Earth Motion (used in ECItoECR, ECRtoECI)

IAU 1980 Nutation Model accepted without correction for IERS/USNO "EOP" data.

Plane Geometric equation used for Equation of the Equinoxes following IAU/USNO practice

## 9.6 Zenith Angle

Simple spherical Earth based on WGS84 hard wired in for parallax correction; saves access to AA data base, subfunction calls, etc.

Accuracy: The largest correction for parallax is for the moon, which is up to one degree, but the error due to use of different Earth models is only due to the variation among them of less than one part in a hundred thousand, which then leads to < 0.1" arc error.

## 9.7 Refraction

The Earth surface gravity "g0" is assumed independent of height and a 1 dimensional spherical atmosphere with constant lapse rate is used.

# Appendix A. File of Leap Seconds (and, for before 1970–other corrections) from the U.S. Naval Observatory

The Toolkit developers have checked by comparing this list with other documents, such as the Astronomical Almanac, that the date in the left column is the beginning of the period when the correction to its right applies. For example, starting Jan. 1, 1977, there were 16 leap seconds, until Jan. 1, 1978. Items after July 1993 (below the dashed line) were added by Science Data Processing Segment (SDPS) Toolkit Developers based on Appendix 3; a table obtained Jan. 30, 1994 from the USNO announcements, series 79, on *tycho.usno.navy.mil* (192.5.41.239). The last column of Appendix A was added by SDPS Toolkit Developers, as shown following the table. Note that the Modified Julian Date MJD is the Julian Date (JD)–2400000.5.

***Table A–1. Leap Seconds (1 of 2)***

| Civil Date | Julian Date | TAI–UTC | Status |
|---|---|---|---|
| 1961 JAN 1 | 2437300.5 | TAI–UTC= 1.4228180S + (MJD–37300.) X 0.001296S | ACTUAL |
| 1961 AUG 1 | 2437512.5 | TAI–UTC= 1.3728180S + (MJD–37300.) X 0.001296S | ACTUAL |
| 1962 JAN 1 | 2437665.5 | TAI–UTC= 1.8458580S + (MJD–37665.) X 0.0011232S | ACTUAL |
| 1963 NOV 1 | 2438334.5 | TAI–UTC= 1.9458580S + (MJD–37665.) X 0.0011232S | ACTUAL |
| 1964 JAN 1 | 2438395.5 | TAI–UTC= 3.2401300S + (MJD–38761.) X 0.001296S | ACTUAL |
| 1964 APR 1 | 2438486.5 | TAI–UTC= 3.3401300S + (MJD–38761.) X 0.001296S | ACTUAL |
| 1964 SEP 1 | 2438639.5 | TAI–UTC= 3.4401300S + (MJD–38761.) X 0.001296S | ACTUAL |
| 1965 JAN 1 | 2438761.5 | TAI–UTC= 3.5401300S + (MJD–38761.) X 0.001296S | ACTUAL |
| 1965 MAR 1 | 2438820.5 | TAI–UTC= 3.6401300S + (MJD–38761.) X 0.001296S | ACTUAL |
| 1965 JUL 1 | 2438942.5 | TAI–UTC= 3.7401300S + (MJD–38761.) X 0.001296S | ACTUAL |
| 1965 SEP 1 | 2439004.5 | TAI–UTC= 3.8401300S + (MJD–38761.) X 0.001296S | ACTUAL |
| 1966 JAN 1 | 2439126.5 | TAI–UTC= 4.3131700S + (MJD–39126.) X 0.002592S | ACTUAL |
| 1968 FEB 1 | 2439887.5 | TAI–UTC= 4.2131700S + (MJD–39126.) X 0.002592S | ACTUAL |
| 1972 JAN 1 | 2441317.5 | TAI–UTC 10.0S + (MJD–41317.) X 0.0S | ACTUAL |
| 1972 JUL 1 | 2441499.5 | TAI–UTC=11.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1973 JAN 1 | 2441683.5 | TAI–UTC=12.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1974 JAN 1 | 2442048.5 | TAI–UTC=13.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1975 JAN 1 | 2442413.5 | TAI–UTC=14.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1976 JAN 1 | 2442778.5 | TAI–UTC=15.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1977 JAN 1 | 2443144.5 | TAI–UTC=16.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1978 JAN 1 | 2443509.5 | TAI–UTC=17.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1979 JAN 1 | 2443874.5 | TAI–UTC=18.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1980 JAN 1 | 2444239.5 | TAI–UTC=19.0S + (MJD–41317.) X 0.0 | ACTUAL |

*Table A–1. Leap Seconds (2 of 2)*

| Civil Date | Julian Date | TAI– UTC | Status |
|---|---|---|---|
| 1981 JUL 1 | 2444786.5 | TAI–UTC=20.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1982 JUL 1 | 2445151.5 | TAI–UTC=21.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1983 JUL 1 | 2445516.5 | TAI–UTC=22.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1985 JUL 1 | 2446247.5 | TAI–UTC=23.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1988 JAN 1 | 2447161.5 | TAI–UTC=24.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1990 JAN 1 | 2447892.5 | TAI–UTC=25.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1991 JAN 1 | 2448257.5 | TAI–UTC=26.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1992 JUL 1 | 2448804.5 | TAI–UTC=27.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1993 JUL 1 | 2449169.5 | TAI–UTC=28.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1994 JUL 1 | 2449534.5 | TAI–UTC=29.0S + (MJD–41317.) X 0.0 | ACTUAL |
| 1996 APR 1 | 2450174.5 | TAI–UTC=30.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 1996 JUL 1 | 2450265.5 | TAI–UTC=31.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 1998 JAN 1 | 2450814.5 | TAI–UTC=32.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 1999 JAN 1 | 2451179.5 | TAI–UTC=33.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2000 JAN 1 | 2451544.5 | TAI–UTC=34.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2001 APR 1 | 2452000.5 | TAI–UTC=35.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2002 APR1 | 2452365.5 | TAI–UTC=36.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2003 APR 1 | 2452730.5 | TAI–UTC=37.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2004 APR 1 | 2453096.5 | TAI–UTC=38.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2005 APR 1 | 2453461.5 | TAI–UTC=39.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2006 APR 1 | 2453826.5 | TAI–UTC=40.0S + (MJD–41317.) X 0.0 | PREDICTED |
| 2007 APR 1 | 2454191.5 | TAI–UTC=41.0S + (MJD–41317.) X 0.0 | PREDICTED |

# Appendix B. Julian Day Conversions with USNO Software

This discussion is intended to clarify the correspondence between Gregorian dates and Julian dates. It refers to standard utilities available from the U.S. Naval Observatory in C to convert among Gregorian and Modified Julian dates. The program names are from USNO, and the programs can all be obtained by ftp or e–mail from that office. The author is indebted to USNO for the original software, which was stripped down to remove the user interface and certain safety checks as it is in incorporated in EOSDIS software that contains other checks. The original algorithms are based on work by Fliegel and van Flandern (*Communications of the Association for Computing Machinery*, Vol. 11, #10. p. 657, Oct. 1968), and the original software was written by Dr. Gernot M. R. Winckler. In detail the algorithms are as follows:

## B.1 Conversion Of Julian Day To Calendar Components

In the following algorithm, all operations are done as C integer operations, and C long integers are used for variables and constants where appropriate. L and n are scratch variables, and the asterisk is used for multiplication, and the equals sign indicates a replacement operation:

```
L = julianDayNumber + 68569
n = 4 *L/146097
L = L- (146097*n + 3)/4
year = integer part of 4000*(L + 1)/1461001
L= L - 1461* year/4 + 31
month = integer part of 80*L/2447
day = integer part of L - 2447 * month/80
L = month/11
month = integer part of month + 2 -12*L
year = integer part of 100*(n - 49) + year + L
```

## B.2 Conversion Of Julian Day To Calendar Components

The same rules apply as in Section B.1; j1, j2, and j3 are long integer scratch variables.

```
j1 = 1461*(year + 4800 + (month - 14)/12)/4
j2 = 367*(month - 2 - (month - 14)/12*12)/12
j3 = 3*((year + 4900 + (month - 14)/12)/100)/4
```

Julian day = (day - 32075 + j1 + j2 - j3)

## B.3 Date and Modified Julian Day Numbers

An example, the following diagrams are given Notation: (JD = Julian Day), UT indicates normal midnight as on a civil clock (for example UTC; but beware, if UTC or UT1 is expressed as a Julian Day, then it starts at noon). Day boundaries are shown as vertical bars. This section explains with charts how the Julian Day (JD) and Modified Julian Day (MJD) boundaries relate to each other and too civil (Gregorian, UTC) dates. Figure B–1 illustrates the relation to Gregorian dates. In this case, there are no minor differences of a few seconds involved—the MJD changes at midnight, UTC and the JD at noon. The only peculiar event is the one second suspension of incrementing the MJD and the JD during the leap second, which the Toolkit software implements in order to access the data tables at the correct place. The suspended portion is restored when appropriate *after* accessing the tables.

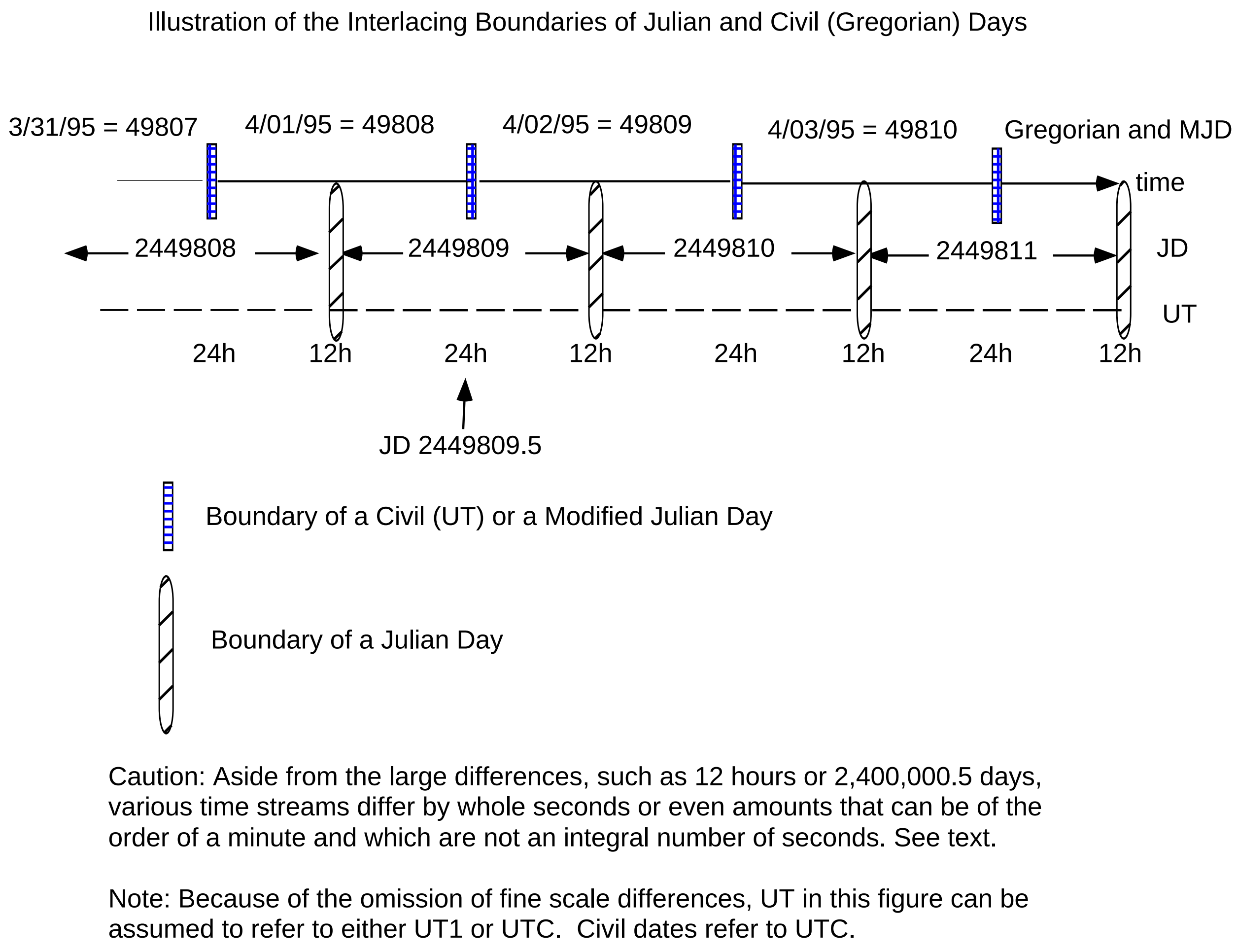


Illustration of the Interlacing Boundaries of Julian and Civil (Gregorian) Days
3/31/95 = 49807
4/01/95 = 49808
4/02/95 = 49809
4/03/95 = 49810
Gregorian and MJD
time
2449808
2449809
2449810
2449811
JD
UT
24h
12h
24h
12h
24h
12h
24h
12h
JD 2449809.5
Boundary of a Civil (UT) or a Modified Julian Day
Boundary of a Julian Day
Caution: Aside from the large differences, such as 12 hours or 2,400,000.5 days, various time streams differ by whole seconds or even amounts that can be of the order of a minute and which are not an integral number of seconds. See text.
Note: Because of the omission of fine scale differences, UT in this figure can be assumed to refer to either UT1 or UTC. Civil dates refer to UTC.

***Figure B–1. Julian, Modified Julian and Gregorian Day Numbers***

- Note that JD 2449808.5, for example, is at midnight, March 31/April 1, while 2449809.5 is at midnight April 1/April 2.

The algorithms The USNO instructions on the program "jdoy," which takes JD to Gregorian date, may benefit from some amplification. It is possible by experimentation to see that the date produced is that on ***whose noon*** in the Julian Day ***began***, or, in other words, the date that ***ends in the middle of that Julian Day***. Thus, entering 2449809 yields 1995 4 1 (i.e., April 1, 1995). Note that April 1 is identical to MJD 48908, but its afternoon overlaps JD 2449809. That is the basis of the remarks in the USNO code that MJD will appear to be one less "without regard" to the fact that JD begins at noon. What is meant is that the MJD continues into the next JD, which starts at noon. So for the afternoon of the civil day, JD increments and the integral MJD is the integral JD minus 24400001, not 24400000. In the Astronomical Almanac, JD 2449808 is listed as April 0, 1995. This corresponds to its having started on the previous noon, and being in progress what Apr. 1 starts. Another way to look at this oddity is to add 0.5, so that 2449808.5 can be identified as the start of Gregorian day Apr. 1. This explains the use of the April 0 notation—you still have to add 1/2 to 2449808 to reach the start of Apr. 1.

The USNO instructions on the program "doymjd," which takes calendar dates to MJD, state that the date is converted to the "full MJD," but implies that the MJD may come out 1 less than the corresponding JD. The MJD is, of course, correct, because Gregorian and Modified Julian dates both start at midnight. What is meant is that in the middle of the MJD, at noon, the JD increments by one. Thus, in the morning, one gets the JD by adding 24400000 to MJD; but in the afternoon, one adds 24400001. This can be checked in the foregoing chart. It was verified by using "doymjd" with the input "04 01 1994," which was converted to 49808. To get the Julian day already in progress at the start of that day, one adds 24400000. At noon on that day, the next JD starts, so in the afternoon one must add 24400001, getting 2449809.

Let's summarize all this for Toolkit user purposes:

> To go from Gregorian calendar day to JD, one must use "doymjd" first to get the MJD. Then if the time was morning, (up to 11:59:59.999999) one must add 24400000 to the MJD (and save the fraction past the previous noon), while if the time is in the afternoon, one must add 24400001. In any case, the hours, minutes, seconds, etc., **past noon** must then be added back in, **in units of days.** Note: Julian days for UT1, UTC, TAI and TDT will roll over at different times. For example, the TAI Julian Day March 1, 1993, began 27 seconds before the UTC Julian day, because there were 27 leap seconds then, while the TDT Julian Day started another 32.184 seconds earlier.

This page intentionally left blank.

# Appendix C.  The File *utcpole.dat*

Following these remarks there are shown the beginning and end of the file "utcpole.dat," as of Dec. 1994, and a section around the June 30, 1993, leap second (MJD 49168/49169). The first column is the MJD; the next four are polar motion data explained in the section on Earth Orientation; the sixth is UT1–UTC; the seventh the error in UT1–UTC; and the eighth the status. The values of UT1–UTC are given daily and are interpolated in between times by the Toolkit. The value jumps upward by about 1 second at leap seconds, because UTC jumps downwards there; the difference is not exactly one whole second because some of the normal variation in UT1 is included (variations in Earth rotation). This can be seen in the middle section of the extract from the original table. The Toolkit automatically "demotes" the value at the next tabulated time by decreasing it 1 second when interpolating; thus, for example, if the time is between MJD 49168 and 49169, where the value jumps up, the value 0.598839 is replaced by a scratch variable -0.401161 s before interpolation. The difference between 0.399545s and 0.401161s is a true measure of the variation in Earth rotation in the elapsed day.

```
MJD         x       err in x         y        err in y      UT1-UTC  err    in    UT1-UTC
status

File Updated: July 14, 1994 by: Peter Noerdlinger using: iers Bulletinb.73,74,75,76

44054  -0.129605    0.001385     0.338255     0.000446     0.084266  0.000117  f

44055  -0.128754    0.001385     0.340346     0.000446     0.082044  0.000117  f

44056  -0.127918    0.001385     0.342434     0.000446     0.079776  0.000117  f

44057  -0.127080    0.001385     0.344523     0.000446     0.077485  0.000117  f

44058  -0.126225    0.001385     0.346615     0.000446     0.075197  0.000117  f

.

.

.

49161  -0.052710    0.000097     0.200525     0.000103     -0.384572  0.000022 f

49162  -0.054053    0.000099     0.201529     0.000110     -0.386732  0.000022 f

49163  -0.055194    0.000100     0.202611     0.000112     -0.389013  0.000024 f

49164  -0.056259    0.000101     0.203723     0.000113     -0.391350  0.000024 f

49165  -0.057352    0.000048     0.204833     0.000076     -0.393650  0.000028 f

49166  -0.058538    0.000110     0.205915     0.000078     -0.395814  0.000034 f
```

```
49167  -0.059810    0.000109     0.206996     0.000075     -0.397775  0.000033 f
49168  -0.061104    0.000107     0.208128     0.000540     -0.399545  0.000033 f
49169  -0.062356    0.000105     0.209315     0.000540     0.598839   0.000030 f
49170  -0.063549    0.000106     0.210527     0.000540     0.597348   0.000031 f
49171  -0.064667    0.000104     0.211724     0.000540     0.595940   0.000028 f
49172  -0.065671    0.000700     0.212854     0.000540     0.594557   0.000027 f
49173  -0.066510    0.000076     0.213889     0.000540     0.593148   0.000024 f
49174  -0.067144    0.000081     0.214829     0.000540     0.591674   0.000025 f
49175  -0.067585    0.000082     0.215717     0.000540     0.590134   0.000066 f
49176  -0.067822    0.000088     0.216608     0.000540     0.588544   0.000066 f
.
.
.
49496  0.184200     0.000000     0.286300     0.000000     -0.153610  0.000000 f
49497  0.183400     0.000000     0.284400     0.000000     -0.155940  0.000000 f
49498  0.182300     0.000000     0.282300     0.000000     -0.158130  0.000000 f
49499  0.181200     0.000000     0.280100     0.000000     -0.160280  0.000000 f
49500  0.180000     0.000000     0.277900     0.000000     -0.162450  0.000000 f
49501  0.179000     0.000000     0.275800     0.000000     -0.164700  0.000000 f
49502  0.178200     0.000000     0.274000     0.000000     -0.166930  0.000000 f
49503  0.177300     0.000000     0.272100     0.000000     -0.169210  0.000000 f
49504  0.176400     0.000000     0.270200     0.000000     -0.170840  0.000000 i
49509  0.171000     0.000000     0.261000     0.000000     -0.181560  0.000000 i
49514  0.165200     0.000000     0.251900     0.000000     -0.192000  0.000000 i
49519  0.158600     0.000000     0.243000     0.000000     -0.202060  0.000000 i
49524  0.151300     0.000000     0.234500     0.000000     -0.211820  0.000000 i
49529  0.143400     0.000000     0.226500     0.000000     -0.221320  0.000000 i
49534  0.134900     0.000000     0.219100     0.000000     0.769450   0.000000 i
49539  0.125800     0.000000     0.212300     0.000000     0.760430   0.000000 i
```

```
49544  0.116100     0.000000     0.206200     0.000000     0.751580   0.000000 i
49549  0.105900     0.000000     0.200800     0.000000     0.742860   0.000000 i
49554  0.095200     0.000000     0.196300     0.000000     0.734220   0.000000 i
49559  0.084100     0.000000     0.192500     0.000000     0.725600   0.000000 i
49564  0.072500     0.000000     0.189700     0.000000     0.716940   0.000000 i
49626.50.000000     0.000000     0.000000     0.000000     0.567000  0.000000  p
49718.50.000000     0.000000     0.000000     0.000000     0.303000  0.000000  p
49808.50.000000     0.000000     0.000000     0.000000     0.058000  0.000000  p
49899.50.000000     0.000000     0.000000     0.000000     -0.183000 0.000000  p
49991.50.000000     0.000000     0.000000     0.000000     -0.376000 0.000000  p
50083.50.000000     0.000000     0.000000     0.000000     -0.641000 0.000000  p
50174.50.000000     0.000000     0.000000     0.000000     0.095000  0.000000  p
50265.50.000000     0.000000     0.000000     0.000000     0.848000  0.000000  p
50357.50.000000     0.000000     0.000000     0.000000     0.653000  0.000000  p
.
.
.
54191.50.000000     0.000000     0.000000     0.000000     0.380000  0.000000  p
54282.50.000000     0.000000     0.000000     0.000000     0.126000  0.000000  p
54374.50.000000     0.000000     0.000000     0.000000     -0.077000 0.000000  p
54466.50.000000     0.000000     0.000000     0.000000     -0.353000 0.000000  p
```

This page intentionally left blank.

# Appendix D. Atmospheric Model

The atmospheric model used for refraction calculations is based on the following idealized equations (see Table 6–2 for definitions):

*Barometric Law:*

$dp/d(h) = - \rho * g0$

where d is the differential operator.

Of course, the true gravity varies with height due to the inverse square law, the centrifugal force, and the fact that with increasing height there is a small increment of mass below (the last of these variations is negligible). It varies with latitude due to the oblateness of the Earth and the variation in centrifugal force. Ideally, one ought to use the true gravity g, but we assume g~g0. The next order correction is that with latitude.

*Ideal Gas Law*

pv = RT/MEAN_MOLEC

The value of MEAN_MOLEC was found from (80 * 28.96 + 1.0 * 18.01)/81.0 = molecular weight of dry tropospheric air (based on 1 part water to 80 dry air by number as derived from typical values in: *Astrophysical Quantities*, *3rd Ed.* by C.W. Allen, (London, the Athlone Press, 1973).

## D.1 Mean Adiabatic Lapse Rate

The actual lapse rate in the troposphere is strongly affected by the adiabatic gas law, but once a constant lapse rate is assumed; the adiabatic law is not needed to close the set of equations. Assuming an empirical global mean adiabatic lapse rate in the troposphere, we get

T = SEA_TEMP – TROPORATE * h, h < TROPOPOAUSE

Above the tropopause an isothermal atmosphere was assumed. The refraction is small there anyway, and it is insensitive to the model. It is hoped to replace the lapse rate and tropopause heights with a latitude dependent ones in Toolkit 5 (the rate is sharply higher, and the tropopause lower, at higher latitudes).

If T is replaced by the above expression then in the troposphere we define

tempFac = 1.0 - TROPORATE * h / SEA_TEMP

and integrate the equation

$dp/\rho = g0 * d(h)$

to obtain

$$\text{densFac} = \text{tempFac}^{\Gamma}$$

where

$$\Gamma = (\text{MEAN\_MOLEC} * g0) / ( R * \text{TROPORATE}) -1$$

In the stratosphere, we assume a constant temperature and scale height and so have an exponential atmosphere with the same scale height as at the tropopause.

Caveat: The altitude is used ONLY to obtain the air pressure, which is then used to obtain the surface index of refraction. Users who employ an inflated Earth radius in geolocation should be especially careful to replace any derived altitude with the height in meters above the geoid before calling this function. The altitude ought to be off the geoid, while in other Toolkit algorithms it is off the ellipsoid. In the Toolkit 3 version, the dependence of density on altitude was wrong, but fortunately not grossly so, due to some approximately canceling algebraic errors.

The index of refraction is assumed to be given by

$$\mu = 1.0 + 0.0002905 (\rho/\rho0) = 1.0 + 0.0002905 * \text{densFac}$$

This is not quite consistent with the Clausius–Mossotti relation $(\mu*\mu-1)/(\mu*\mu+2) = \rho*\text{const}$, but agrees within 3.5 parts per billion between -1000 and + 25,000 m altitude

This concludes the atmospheric model. It is hoped to develop simple latitude dependent models of the quantities TROPORATE, TROPOPAUSE, ρ0, and SEA_TEMP by the next increment of the toolkit.

## D.2 Density Scale Height

For the simplified, one layer atmospheric model used to estimate the displacement at small zenith angle, one needs the density scale height, $dh/d(\log(\rho))$. This is easily derived from the previous equations as $dh/d(\log(\rho)) = T /(( \text{MEAN\_MOLEC} * g0 / R) - \text{TROPORATE})$

# Appendix E. Celestial Body Models

***Table E–1. Physical Radii for CB in FOV Tool***

| Celestial Body ID (cbID) | Radius (km) | Explanation |
|---|---|---|
| PGSd_SUN | 7 e 5 | |
| PGSd_MOON | 1739 | allows for topography |
| PGSd_MERCURY | 2440 | |
| PGSd_VENUS | 6055 | |
| PGSd_EARTH | n/a | use tool PGS_CSC_Earthpt_FOV() |
| PGSd_MARS | 3397 | ignore satellites |
| PGSd_JUPITER | 1890 e 3 | includes Galilean satellites |
| PGSd_SATURN | 1225 e 3 | includes rings, satellites to Titan |
| PGSd_URANUS | 25600 | planet only |
| PGSd_NEPTUNE | 24800 | planet only |
| PGSd_PLUTO | 19600 | includes Charon |
| PGSd_CB | 0.0 | a star or user–defined point |

Notes:

We have included satellites down to the 10th magnitude.

Various approximations were made in the Toolkit to streamline it. These will result in negligible degradation of accuracy. Following is a brief list:

- Approximations Made for Efficiency and Speed in the TD, CBP, EPH and CSC portions of the PGS Toolkit

This page intentionally left blank.

# Appendix F.  Conical Hull Test

The two functions PGS_CBP_body_inFOV() and PGS_CSC_Earthpt_FOV() have similar jobs: to find if a point or any portion of a finite disk is in the FOV (field of view) of an instrument. The function described herein has two purposes:

a. it will speed up the tasks by obviating complicated algorithms for points well away from the FOV

b. it will enable detection and rejection of FOV specifications outside our present algorithmic limits. [Present software does not reliably handle fields of view more than 180 degrees across.]

The concept is simple: One draws a circular cone around the FOV, and first checks if the candidate point is inside it before going further. The usage will be somewhat different for PGS_CBP_body_inFOV() and PGS_CSC_Earthpt_FOV() because in the first case the satellite is a finite disk, while in the second, the Earth point is only a point.

This page intentionally left blank.

# Appendix G. Details on the JPL Ephemeris

The version of the ephemeris released on April 30, 1994, is a binary version made from the ASCII version "de200.peter" on the JPL navigator server, utilizing a translation program adapted from the JPL translation program "ascii2eph.f". The present release functions exactly as any of the JPL de200 ephemerides would; only the time span is different, being tailored to the toolkit. "README" documents sent with the ephemeris explain its provenance more fully; readers who wish to access documentation direct from JPL are advised to use anonymous ftp from "navigator.jpl.nasa.gov."

The ephemeris was obtained by ftp from the server navigator.jpl.nasa.gov on April 5, 1994.

***Table G–1. Original Ephemeris Location Information***

| Directory | File Type | length | file name |
|---|---|---|---|
| /ephem/export/ascii | ascii | 17701056 | de200.peter |

Date file stored by JPL: Apr 5 23:51

Data from head of file:

KSIZE= 1652 NCOEFF= 826 (parameters for record structure and interpolation)

JPL Planetary Ephemeris DE200/DE200

Start Epoch: JED= 2433264.5 1949 DEC 14 00:00:00

Final Epoch: JED= 2459215.5 2021 JAN 01 00:00:00

This ephemeris was translated from ascii to binary with the program: asc2eph.f, with the reduced Julian Date range:

T1 = 2436175.5D0

T2 = 2459215.5D0

(Dec 2, 1957 0 hrs to Jan 1, 2021 0 hrs)

The filename used by EOSDIS for this version, in binary, is:

de200.EOS

These ephemerides are used for the retrieval of the ephemeris data:

- Cartesian coordinates of the major planets, moon and sun at any requested time Julian Ephemeris Date (JED) covered by the ephemeris.

- The original ephemeris on which de200.EOS is based is a J2000–based ephemeris: de200, running from JED2305424.5 (1599 DEC 09) to JED 2513360.5 (2169 MAR 31). The EOS version de200.EOS was checked against the JPL test data file "testpo.200" and performed identically with the full de200, with the exception that on the Silicon Graphics International (SGI) platform de200.EOS showed a sprinkling of error of one part in 1e13 where the full ephemeris did not. The error is insignificant, but the matter is under study.
- The original version is so large that it was deemed unwieldy to be sent with the Science Data Processing Segment (SDPS) Toolkit.
- The version supplied by the PGS Toolkit in April 1994 is unrelated other than by direct coding to binary with the JPL program. It is anticipated that in the near future the EOSDIS ephemerides will be corrected for light travel time to the Earth and for aberration due to the Earth's orbital motion. The two corrections are not expected to exceed 1 minute arc.

The original JPL description of DE118 and DE 200 is:

- DE118: created in 1980, was based on the (1950) equator and equinox of the next increment. It was used to create DE200.
- DE200: created from DE118 by rotating all coordinates onto the equator and equinox of J2000 (FK5). This ephemeris has been the basis of the Astronomical Almanac since 1984.

The following references are pertinent:

- Newhall, X X, Standish, E.M. and Williams, J.G.: 1983, "DE102: a numerically integrated ephemeris of the moon and planets spanning forty–four centuries", Astronomy & Astrophysics, vol. 125, pp. 150–167.
- Standish, E.M.: 1982, "Orientation of the JPL Ephemerides, DE200/LE200, to the Dynamical Equinox of J2000", Astronomy & Astrophysics, vol. 114, pp. 297–302.
- Standish, E.M.: 1985, "Planetary and Lunar Ephemerides, DE125/LE125", JPL IOM 314.6–591.
- Standish, E.M.: 1987, "Ephemerides, DE130/LE130 & DE202/LE202", JPL IOM 314.6–891.
- Standish, E.M.: 1990, "The Observational Basis for JPL's DE200, the planetary ephemeris of the Astronomical Almanac", Astronomy & Astrophysics, vol. 233, pp. 252–271.
- Standish, E.M.: 1991, "The JPL Planetary and Lunar Ephemerides DE234/LE234", JPL IOM 314.6–1348.

Identifications for "npl" and "nctr"

| | |
|---|---|
| 1 = mercury | 8 = neptune |
| 2 = venus | 9 = pluto |

| | |
|---|---|
| 3 = earth | 10 = moon |
| 4 = mars | 11 = sun |
| 5 = jupiter | 12 = solar–system barycenter |
| 6 = saturn | 13 = earth–moon barycenter |
| 7 = uranus | 14 = nutations in longitude and obliuity |
| | 15 = librations (if they exist on the file) |

(for nutations and librations, nctr=0)

list(12) [int.]: vector specifying which of the bodies on the file are requested

list(i)=0, no interpolation for body i

=1, position only

=2, position and velocity

the designation of the bodies for 'list' is

list(i) = 1: mercury

= 2: venus

= 3: earth–moon barycenter

= 4: mars

= 5: jupiter

= 6: saturn

= 7: uranus

= 8: neptune

= 9: pluto

=10: geocentric moon

=11: nutations in longitude and obliquity

=12: lunar librations (if on file)

We are indebted to the Jet Propulsion Laboratory, and especially to E. Myles Standish and Faith A. McCreary for their assistance.

This page intentionally left blank.

# Appendix H. Earth Curvature in the Meridian Plane

To prove this, note that the curvature, while it is defined as the rate of turning of the tangent, can just as easily be found from the rate of turning of the normal, which is perpendicular to the tangent. [See D.V. Widder, Advanced Calculus, Prentice–Hall, New York 1947, p 84.] Thus,

curvature_in_meridian = | d(tangent)/d($\phi$) * d($\phi$)/ds | = | d(normal)/d($\phi$) * d($\phi$)/ds |,

where s is arc length. But if we work in the prime meridian (zero longitude) the normal vector is just

normal = [cos($\phi$), sin($\phi$)],

and its derivative with respect to latitude has unit length. Thus,

curvature_in_meridian = | d($\phi$)/ds | = 1.0/|ds/d($\phi$)|.

But ds/d($\phi$) is easily found in the prime meridian plane (using $ds^2 = dr^2 + dZ^2$) by differentiating the expressions for r and Z in the first section. The result is A*(1-ecc2) * $(\xi)^{3/2}$. Using the fact that radius of curvature is the reciprocal of curvature; the proof is complete.

This page intentionally left blank.

# Appendix I. The Barycentric Time Correction for Earthbound Clocks

This Appendix derives the relationship between Barycentric Dynamical Time TDB and Terrestrial Dynamic Time TDT. According to general relativity, *ds*, the line element, is the element of distance in light seconds for spacelike increments $dx^i$ and is increment in time registered in seconds by a standard clock for timelike displacements. By the principle of equivalence, Terrestrial Dynamical Time (TDT), i.e., time as measured by an ideal atomic clock on Earth, is the same as would be measured by the Earth's own rotation in the absence of perturbations, such as tides and geophysical effects. The behavior of such a clock will differ from that of an ideal clock at rest with regard to the barycenter of the Solar System but removed to such a large distance that solar and planetary gravitational effects are negligible. We shall refer to such a distant clock as a "Schwarzschild coordinate clock," measuring a time denoted *t*. Please note that such a clock does *not* measure Barycentric Dynamical Time (TDB) as defined in the *1994 Astronomical Almanac*, even though that time is referred to as a "coordinate time" on p. M2. The reason is that TDB is forced to run at the same *mean* rate as TDT. This means that the average effect of the sun, moon, and planets (including Earth) is included. In the present discussion, the effects of the gravity of the planets, moon and Earth on the clocks is ignored, in which case our coordinate time becomes the same as the IAU coordinate time Barycentric Coordinate Time (TCB). The ignored effects are significant only at the level of less than a part per billion in rate. Also see remarks just before Equation (4), below, and footnote 4. The Schwarzschild coordinate clock runs somewhat faster than TDB, because it is far from the sun's gravitational potential and because it does not partake of the Earth's orbital motion; the two effects are approximately equal (see the discussion following Equation [2]). We shall represent the gravity field of the sun by the Schwarschild exterior metric. To a first approximation, an earthbound clock executes an elliptical path around the sun. There are corrections for the center of mass of the Earth/sun system, the motion of the sun about the Solar System barycenter, planetary and lunar perturbations, and the Earth's rotation and gravitational potential, all of which we'll ignore for the moment. We'll use the motion of the Earth–moon barycenter as that of a clock on the Earth and will use *ds* for the element of proper time along the world line of the clock.

If *M* is the mass of the sun and *v* the speed of the Earth in its orbit, then to first order in *M* and second order in *v/c* (ignoring products of *GM* and $v^2$ ) we have[10]

$$\frac{ds}{dt} = 1 - \frac{GM}{rc^2} - \frac{v^2}{2c^2} \tag{1}$$

[10]H. P. Robertson and T. W. Noonan, *Relativity and Cosmology* (W. B. Saunders, New York, 1968) p. 259.

where $G$ = Newton's constant of gravity and $r$ is the distance from the Earth to the sun. To the order considered, it does not matter if we measure $r$ and $v$, for example, in standard Schwarzschild coordinates, isotropic Schwarzschild coordinates, etc. A separate calculation was done (see below) taking into account the mass of the Earth compared to that of the sun with a Newtonian treatment, but I ignored the perturbations of the other planets. This is reasonable for studying clock rate variations during a year, because of the long orbital period of the major perturber (Jupiter). The increment in the reading of a standard Earth bound clock is thus

$$\Delta s = \int \frac{ds}{dt} dt = \int \frac{ds}{dt}\frac{dt}{d\theta} d\theta \tag{2}$$

where $d\theta$ is some measure of angular motion around the sun. Note that $\theta$ need not (and will not) be a true angle in the geometric sense. For some purposes we would do best to take $\theta$ to be the mean anomaly, because that is the proper argument for certain trigonometric functions that appear below. This choice would amount to using the anomalistic year of 365.259635 days, each of 86400 SI seconds of TDT to define the period; but use of the mean anomaly would complicate the representation of Kepler's third law, as it relates to the definition of the Solar Gaussian gravitational constant, $GM$. Therefore, we take $d\theta$ to have a rate yielding a full circle in one *sidereal* year, 365.256363 days. The difference is only in the 6th significant digit. The correction is needed here to only one or two digits, while the discrepancy with the Astronomical Almanac is in the fourth digit. According to the Virial Theorem[11] the following equation holds for the mean values of $GM/r$ and $v^2$:

$$\left\langle \frac{GM}{r} \right\rangle = \frac{GM}{a} = \left\langle v^2 \right\rangle \tag{3}$$

According to the conservation of energy, for a negligible mass orbiting the sun, the variations in last two terms in Equation (1) are equal, so that for variations, we may take one of them doubled in place of the two (we choose the term in 1/r). In terms of equations, we may then write for the steady part and the varying part (indicated by the symbol Δ):

$$\frac{GM}{r} + \frac{v^2}{2} = \left\langle \frac{GM}{r} + \frac{v^2}{2} \right\rangle + \Delta\left( \frac{GM}{r} + \frac{v^2}{2} \right) = \frac{3}{2}\frac{GM}{a} + 2\Delta\left( \frac{GM}{r} \right) \tag{4}$$

This substitution, which eliminates $v$ in favor of $r$, is based on a fixed sun. It is possible to show that the corrections to account for the finite mass ratio of sun and Earth is only a few parts per million of the correction terms themselves. Only the variation in Equations (1) or (4) is important, because a constant simply renormalizes the scale of $dt$. A standard clock at "infinity" would actually run slightly faster than a "smoothed" Earthbound one, the ratio being the reciprocal of the mean value of Equation (1). Although this rate difference is unobservable,

[11]S. Goldstein, *Classical Mechanics,* Addison-Wesley 1950, p. 69.

because we do not have access to an atomic clock beyond, say, the orbit of Pluto, the IAU has recently defined “Barycentric Coordinate Time” in an attempt to calibrate to such a hypothetical clock (also see footnote 4).

If $a$ is the astronomical unit, the angle θ (essentially the mean anomaly) increases at the rate

$$\frac{d\theta}{dt} = \frac{\sqrt{GM}}{a^{3/2}} \tag{5}$$

Using a well known relation from Escobal[12], e.g., we find

$$\frac{1}{r} = \frac{1}{a}[1 + 2 \sum_{m=1}^{\infty} J_m(me)\cos(m\theta)] \tag{6}$$

where $J_m$ is the $m$’th order Bessel function. Substituting Equations (2 – (6) in Equation (1) and integrating, to get the full excursion of TDB from TDB, we find

$$\Delta s = \frac{a^{3/2}}{GM} \int d\theta \left\{ 1 - \frac{3}{2}\frac{GM}{ac^2} - 4\frac{GM}{ac^2} \sum_{m=1}^{\infty} J_m(me)\cos(m\theta) \right\} \tag{7}$$

where $e$ is the eccentricity of the Earth's orbit. The perturbations of the planets cause $e$ to vary over time spans of years to centuries. Some decades ago, the value of $e$ as listed in Allen's *Astrophysical Quantities* was 0.016722. During 1994, however, the *Astronomical Almanac* gives values about 0.01667 - 0.01668, quite a bit less. The steady term simply totals up TDB, while the periodic terms yield the difference TDT–TDB. Strictly speaking, we ought to normalize the periodic terms by dividing by the ratio

$$1 - \frac{3GM}{2ac^2} \approx 1 - 1.48 \times 10^{-8}, \tag{8}$$

but we need the periodic terms only to one or two significant figures for the Toolkit, or to four significant figures at the most, to compare with the Astronomical Almanac. To keep this correction would be essentially to keep the difference in TDB and TCB[13]. Discarding the correction in Equation (8) and saving only the periodic terms yields

[12]P.R. Escobal, *Methods of Orbit Determination* (John Wiley & Sons, New York, 1965) pp. 85-87, eqs. (3.67), (3.81), (3.83) and (3.89)

[13]There is an additional correction of order 7x10$^{-10}$ in rate that the IAU uses to try to take out the effect of the geopotential. Note, by the way, that rate discrepancies cumulate (they cause secular deviation of the underlying time streams).

$$TDT - TDB = -4\frac{a^{3/2}}{\sqrt{GM}}\frac{GM}{ac^2}\sum_{m=1}^{\infty} J_m(me)m^{-1}\sin(m\theta) \tag{9}$$

The factor in $a^{3/2}/(GM)^{1/2}$ may be identified from Kepler's Third Law as the number of seconds in a year, divided by 2π. After integration the series in Equation (7) will be of the form

$$TDT - TDB = \sum_{m=1}^{\infty} a_m \sin(m\theta) \tag{10}$$

After evaluation of the Bessel functions, the following results are obtained for different values of *e:*

***Table I–1. Coefficients for the sine series for TDT–TDB for difference values of e***

| ***e*** | ***m*** | **coefficient *$a_m$* of sin(*$m\theta$*) (sec)** |
|---|---|---|
| 0.166700 | 1 | 0.0016528 |
| . | 2 | 0.0000138 |
| . | 3 | 0.0000002 |
| 0.0166750 | 1 | 0.0016533 |
| . | 2 | 0.0000138 |
| . | 3 | 0.0000002 |
| 0.0167220 | 1 | 0.0016580 |
| . | 2 | 0.0000139 |
| . | 3 | 0.0000002 |

The last set matches the USNO values on p. B5 of the *1995 Astronomical Almanac* (only the first two coefficients are given there), but one of the first two sets of values would seem better. The slight discrepancy between our results and the Astronomical Almanac that emerges is not significant at our level, but the Toolkit software uses the Almanac value anyway. Recalling that TDB is needed only for the DE200 ephemeris and the nutation routine; one sees that corrections of less than a millisecond are unimportant. The leading term in the total correction has amplitude only ~ 1.7 milliseconds, so we need it to only to two significant figures in the most conservative case.

# Appendix J.  Listing of SDP Toolkit Geolocation Tools

In this Appendix, we list the SDP Toolkit calls referenced in the text in this document. These calls will be implemented in their entirety by the Feb. 95 incremental software delivery. It is expected that upon EOS community usage of this package, the tools will be enhanced, efficiency increased, functionality added in concurrence with ESDIS, and so on.

***Table J–1.  Toolkit Routine Key***

| Key | Class |
|---|---|
| CBP | Celestial Body Position |
| CSC | Coordinate System Conversion |
| EPH | Ephemeris Data Access |
| TD | Time Date Conversion |

***Table J–2.  Toolkit Routine Listing (1 of 2)***

| Tool Name | Description |
|---|---|
| PGS_CBP_body_in_FOV | Given instrument parameters, returns a flag to indicate whether any of the user–selected major celestial bodies (sun, moon, etc.) are in the instrument field–of–view. |
| PGS_CBP_BrightStar_ positions (Toolkit 5–under negotiation) | Returns the position of all stars of magnitude > input magnitude, or the position of the stars designated by inputting star id's and setting the flag 'input_flag'. |
| PGS_CBP_Earth_CB_Vector | Computes the ECI frame vector from the Earth to the sun, moon, or planets at a given time, or range of time(s). |
| PGS_CBP_Sat_CB_Vector | Computes the ECI vector from the spacecraft to the sun, moon, or planets at a given time or range of time(s) |
| PGS_CBP_SolarTimeCoords | Computes local solar time, and right ascension and declination of the sun, for a given standard time and position on the surface of the Earth. |
| PGS_CSC_DayNight | Determines whether a given point on the Earth is in day, night or twilight, at a given time. |
| PGS_CSC_Earthpt_FOV | For a field of view defined by a table of coordinates (accessed externally), and a known motion of the boresight vector as a function of time, obtains the UTC time interval and the starting time that an Earth point is within the field–of–view, within a specified time window. |
| PGS_CSC_ECItoECR | Coordinate system conversion |
| PGS_CSC_ECItoSC | Coordinate system conversion |
| PGS_CSC_ECRtoECI | Coordinate system conversion |
| PGS_CSC_ECRtoGEO | Coordinate system conversion |
| PGS_CSC_GEOtoECR | Coordinate system conversion |

***Table J–2.  Toolkit Routine Listing (2 of 2)***

| Tool Name | Description |
|---|---|
| PGS_CSC_GetFOV_Pixel | Computes the projection of (geolocates) the instrument field–of–view on the Earth, optionally, geolocates the center of each pixel in the footprint. |
| PGS_CSC_GreenwichHour | Returns the Greenwich Hour Angle of the vernal equinox, which is equal to Greenwich sidereal time, in the ECI frame, at a given time. |
| PGS_CSC_ORBtoSC | Frame change tool |
| PGS_CSC_SCtoECI | Frame change tool |
| PGS_CSC_SCtoORB | Frame change tool |
| PGS_CSC_SubSatPoint | Returns the position and velocity vector of the sub–satellite point ("pierce point"), or nadir of the satellite on the Earth's surface. Optionally returns the nadir vector also. |
| PGS_CSC_ZenithAzimuth | Returns zenith and azimuth angles of spacecraft. |
| PGS_TD_ASCIItime_AtoB | Converts binary time values to ASCII Code B time values of the form year_month_day_time_of_day in the CCSDS format. |
| PGS_TD_ASCIItime_BtoA | Converts binary time values to ASCII Code A time values of the form year_month_day_time_of_day in the CCSDS format. |
| PGS_TD_GPStoUTC | Converts to UTC time value from  GPS time by converting to internal time, adding the GPS_minus_UTC_leapseconds from the leapseconds file, and converting to GPS format following CCSDS ASCII standard A[14]. |
| PGS_TD_SCtime_to_UTC | Converts spacecraft clock time to UTC for EOS platforms or for foreign spacecraft. |
| PGS_TD_TAItoUTC | Converts TAI time value to UTC time. |
| PGS_TD_TimeInterval | Computes the elapsed TAI time in seconds between any two epochs after January 1, 1958. |
| PGS_TD_UTCtoGPS | Converts UTC time value to GPS time by converting to internal time, adding the GPS_minus_UTC_leapseconds from the leapseconds file, and converting to GPS format following CCSDS ASCII standard A (see footnote). |
| PGS_TD_UTCtoTAI | Converts UTC time to TAI time by first converting UTC to internal time and them adding the TAI_minus_UTC_leapseconds from the leapseconds file. |
| PGS_TD_UTCtoTDBjed | UTC to TDB time conversion |
| PGS_TD_UTCtoTDTjed | UTC to TDT time conversion |
| PGS_TD_UTCtoUT1 | Converts UTC to UT1 time. |
| PGS_TD_UTC_to_SCtime | Converts UTC to Spacecraft clock time for EOS standard of Foreign Spacecraft. |

[14]In TK5 the GPS time will be re-referenced to Jan 6, 1980

# Abbreviations and Acronyms

| | |
|---|---|
| A.A. | Astronomical Almanac |
| AAS | American Astronautical Society |
| ACS | Attitude Control System |
| AIAA | American Institute of Aeronautics and Astronautics |
| API | application program interface |
| ASCII | American Standard Code for Information Interchange |
| ATBD | algorithm theoretical basis document |
| BIH | Bureau International de l'Heure |
| CB | Celestial Body |
| CBP | Celestial Body Position |
| CCSDS | Consultative Committee for Space Data Systems |
| CDS | CCSDS Day Segmented |
| CERES | Clouds and Earth’s Radiant Energy System |
| COSMIC | Computer Software Management and Information Center |
| CSC | Coordinate System Conversion |
| CUC | CCSDS Unsegmented Time Code |
| cy | century |
| DAAC | Distributed Active Archive Center |
| DEC | Declination |
| DEM | digital elevation model |
| ECI | Earth Centered Inertial |
| ECR | Earth Centered Rotating |
| ECS | EOSDIS Core System |
| EDF | ECS Development FAcility |
| EOP | Earth Orientation Parameters |
| EOS | Earth Observing System |
| EOSAM | EOS AM Project (morning equator crossing spacecraft series) |
| EOSDIS | Earth Observing System Data and Information System |

| | |
|---|---|
| EOSPM | EOS PM Project (afternoon equator crossing spacecraft series) |
| EPH | Ephemeris Data Access |
| ESDIS | Earth Science Data and Information System |
| ET | Ephemeris Time |
| FDF | Flight Dynamics Facility |
| FDSS | Flight Dynamic Support System |
| FK5 | Fundamental Catalog 5 |
| FOS | Flight Operations Segment |
| FOV | Field of View |
| ftp | file transfer protocol |
| GAST | Greenwich Apparent Sidereal Time |
| GMST | Greenwich Mean Sidereal Time |
| GPS | Global Positioning System |
| GSFC | Goddard Space Flight Center |
| GST | Greenwich Apparent Sidereal Time (in the IERS Standards) |
| IAU | International Astronomical Union |
| IERS | International Earth Rotation Service |
| JD | Julian Date |
| JED | Julian Ephemeris Date |
| JPL | Jet Propulsion Laboratory |
| LAGEOS | Laser Geodynamics Satellite |
| LEO | Low Earth Orbiting |
| l.o.d. | length of the day |
| m | meter |
| MISR | Multi–Angle Imaging SpectroRadiometer |
| MJD | Modified Julian Date |
| MODIS | Moderate–Resolution Imaging Spectrometer |
| NIST | National Institute of Standards and Technology |
| NOAA | National Oceanic and Atmospheric Administration |
| NSP | North–South Plane |
| PGS | Product Generation System |

| | |
|---|---|
| RA | Right Ascension |
| RDC | Research Data Corporation |
| s | second |
| SC | Space Craft |
| SDP | Science Data Processing |
| SDPS | Science Data Processing Segment |
| SGI | Silicon Graphics International |
| SI | Systeme International |
| SMF | Status Message File |
| TAI | Temps Atomique International |
| TBD | To Be Determined |
| TCG | Geocentric Coordinate Time |
| TDB | Barycentric Dynamical Time |
| TDRSS | Tracking and Data Relay Satellite System |
| TDT | Terrestrial Dynamical Time |
| TOD | True of Date |
| TRMM | Tropical Rainfall Measuring Mission |
| TSDIS | TRMM Science Data and Information System |
| TT | Terrestrial Time |
| UARS | Upper Atmosphere Research Satellite |
| USAF | United States Air Force |
| USNO | United States Naval Observatory |
| UT0 | Universal Time (uncorrected for polar motion) |
| UT1 | Universal Time (corrected for polar motion) |
| UTC | Coordinated Universal Time |
| VLBI | Very Long Baseline Interformetry |