LA-UR-16-21285

Approved for public release; distribution is unlimited.

Title: Solar Position Model for use in DIORAMA

Author(s): Werley, Kenneth Alan

Intended for: Report

Issued: 2016-03-01

Solar Position Model for use in DIORAMA

Kenneth A Werley

14 October 2014

1 Introduction

The DIORAMA code requires the solar position relative to earth in order to compute GPS satellite orientation. The present document describes two functions that compute the unit vector from either the center of the Earth to the Sun or from any observer's position to the Sun at some specified time. Another function determines if a satellite lies within the Earth's shadow umbra. Similarly, functions determine the position of the moon and whether a satellite lies within the Moon's shadow umbra.

2 Calculation of Earth-Sun Unit Vector

The unit vector pointing from the earth to the sun, given in rectangular ICRF coordinates, is computed for a specified date/time input. The time input is given in Boost library boost::posix_time::ptime format that consists of a gregorian date, *e.g.*, *date(year,month,day)*, together with a time duration of that day, *e.g.*, *time_duration(hours, minutes, seconds, fractional_seconds)*.

The computation of the Earth-Sun consists of four major steps. The first step converts the time to the fractional number of days, d, since Greenwich noon, Terrestrial Time, 1/1/2000.[1] The second step computes the obliquity of the ecliptic, $\epsilon = 23.439 - 0.0000004d$.[1] The third step computes the ITRF solar position in spherical coordinates. Two options for the third step includes either: (a) a Low Accuracy Model[1], or (b) a High Accuracy Model[2]. Two options are provided for variation in runtime, accuracy, and for comparison testing. The High Accuracy Model accounts for the moon and other planet gravitational perturbations on the Earth's orbit. The final step converts the result to a rectangular-corrordinate (x, y, z) unit vector.

The low accuracy model runs in about a tenth of the time as the high accuracy model, or about 1 μs *vs.* 10 μs on one test computer. The high accuracy model runs sufficiently fast that it is set as the default option for DIORAMA.

While suffering a runtime penalty, the high accuracy model achieves the following benefit in accuracy. Example of low and high accuracy Sun position results:

Solar position (spherical coordinates)
  Sun lon=-169.5092513 lat=-4.5130392 [deg]   Low Accuracy
  Sun lon=-169.5019489 lat=-4.5160498 [deg]   High Accuracy

Satellite-Earth-Sun beta angle:
  beta=-0.078767 [rad]   Low Accuracy
  beta=-0.078820 [rad]   High Accuracy

Differences occur in the third digit past the decimal for a decimal degree result. The solar position is used in DIORAMA to compute the Satellite-Earth-Sun beta angle. A sample computed beta angle in radians varies by less than 0.01% between the Low and High models.

3 Sample Results for Solar Position:

3.1 Sample Results of Interim Steps

Table I demonstrates results from the four interim steps applied in computing the Earth-Sun unit vector. The final Step 4 illustrates the final result that may be used for computations of GPS satellite orientation.

Table I. Sample Results of Interim Steps

```
Input:    time = 10/4/2014 17:30:03.14

Step 1: fractional days since 1/1/2000 is d = 5390.229 [days]
Step 2: obliquity of the ecliptic is    eps = 23.437 [deg]
Step 3: solar position (lon,lat) = (-169.502,-4.516) [deg]
Step 4: Earth-Sun unit vector:
             x=-0.9802, y=-0.1816, z=-0.0787
```

3.2 Sample Results of Solar Position at Five Times of the Year

In order to demonstrate solar movement, Table II lists the Earth-Sun unit vector computed at five times of the year at three-month intervals. These results oscillate every six months, as expected. The oscillation in x from month 4 to month 10 illustrates the sun moving above and below the equator. After a full year, the sun returns to nearly its original position.

Table II. Sample Results at Five Times of the Year

```
time= 1/01/2014 12:00:00   unit vector:  x= 0.1907, y=-0.9007, z=-0.3904
time= 4/01/2014 12:00:00   unit vector:  x= 0.9793, y= 0.1858, z= 0.0805
time= 7/01/2014 12:00:00   unit vector:  x=-0.1666, y= 0.9047, z= 0.3922
time=10/01/2014 12:00:00   unit vector:  x=-0.9897, y=-0.1315, z=-0.0570
time= 1/01/2015 12:00:00   unit vector:  x= 0.1864, y=-0.9014, z=-0.3908
```

4 Calculation of Observer-Sun Unit Vector

The unit vector pointing from an observer to the sun, given in rectangular ICRF coordinates, is computed for a specified date/time input and an observer position given in ICRF coordinates. The time input is given in Boost library boost::posix_time::ptime format.

The computation of the Observer-Sun unit vector requires three steps. First, the distance from the Earth to the Sun is computed[1]. Second, the Earth-to-Sun unit vector is computed using the function described above. The Earth-Sun distance is applied to un-normalize the Earth-Sun unit vector and provide a sun position. The final step subtracts the observer position from the sun position and then renormalizes by the new vector's magnitude to construct a new Observer-Sun unit vector. The final result is similar to that for the Earth-Sun unit vector.

5 Example Usage of DIORAMA Function for Accessing the Sun's Position

The function, *SunPositionFromEarth(t,sun,1)*, computes the unit vector from the center of the Earth to the Sun. Below is a *SunPositionFromEarth()* function description followed by sample DIORAMA code that calls the function.

```
/* function  SunPositionFromEarth
* @brief Compute the earth-to-sun unit vector at a specified time.
* @param[in]      t                The date/time.
* @param[in]   highAccuracy    1=highly Accurate sun position; 0=rough.
* @param[in]    iprt   Print level: 0=nothing; 1=results.
* @param[out]   sun                The earth center to sun to unit vector.
*/

// example calculating sun unit vector:
boost::posix_time::ptime t7(boost::gregorian::date(2002,3,10),            //set the date/time
                            boost::posix_time::hours(1)+boost::posix_time::minutes(2)+
                            boost::posix_time::seconds(3)+boost::posix_time::milliseconds(4)+
                            boost::posix_time::nanosec(5));
PosITRF sun;
diorama_solar ds;

ds.diorama_solar::SunPositionFromEarth(t7,sun,1,0);

cout«"SunPositionFromEarth: unit postion (x,y,z)=("«sun.x«","«sun.y«","«sun.z«") \n";
```

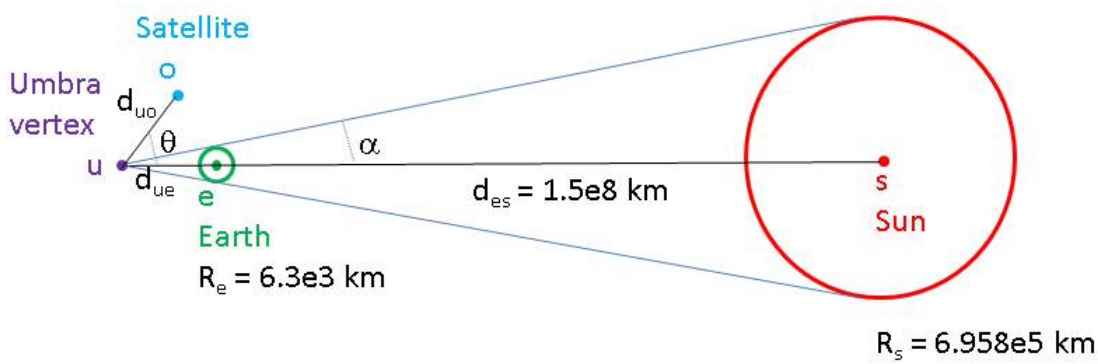
6 Determination if a Satellite lies within the Earth's Umbra

Figure 1 illustrates the geometry applied to compute if a satellite lies within the shadow of the Earth. The Earth and Sun are both approximated to be spherical. Initial (known) required information includes the satellite position at a specified time. Then, the first step uses the routine described above to compute the position of the Sun and its distance from the Earth. Step 2 computes the umbra vertex position and angle α by applying the equations listed in Fig. 1. Step 3 computes the distance from the vertex to the satellite and the associated angle θ. Finally, if the vertex-to-satellite distance is less than the vertex-to-Earth distance and $\theta < \alpha$, then the satellite lies within Earth's umbra.

A second algorithm, suggested for use in satellite orientation models[3], for estimating when a satellite moves into the Earth's shadow, treats the Sun as a point, and applies a reduced, effective Earth's radius of $0.9818R_e$. The value of the effective radius is chosen so that, for a GPS satellite orbital radius of about 26,500 km, the shadow entry agrees with the Umbra model. This second algorithm was tested and discarded because it is incorrect at other satellite orbital radii.

The function, *IsInUmbraShadow(t,***o***)*, returns a *true* if a satellite lies in the Earth's shadow. The input parameters specify the time, t, and the satellite ITRF orbital position, **o**.

Figure 1. Does a Satellite lie within the Earth's Shadow ?



Algorithm Steps:

0. know **o** & time, t
1. Calc **s** & d_{es} [SunPositionFromEarth()]
2. Calc d_{ue} , **u** , α [$d_{ue} = d_{es} R_e/(R_s - R_e)$; $\mathbf{u} = - d_{ue}\, \mathbf{s}_u$; $\sin \alpha = R_e / d_{ue}$]
3. Calc d_{uo} & θ [$\mathbf{o.u} = ou \cos\gamma$; $\sin\theta = d_{oe} \sin\gamma / d_{ue}$]
4. *If* ($d_{uo} < d_{ue}$ and $\theta < \alpha$ *), in shadow*

Final Function: *tle::IsInUmbraShadow(t,o)* Assumption: *Spherical Earth*

7 Calculation of Earth-Moon Unit Vector

The unit vector pointing from the earth to the moon, given in rectangular ICRF coordinates, is computed for a specified date/time input. The time input is given in Boost library boost::posix_time::ptime format that consists of a gregorian date, *e.g.*, *date(year,month,day)*, together with a time duration of that day, *e.g.*, *time_duration(hours, minutes, seconds, fractional_seconds)*.

The computation of the Earth-Moon vector consists of four major steps. The first step converts the time to the fractional number of days, d, since Greenwich noon, Terrestrial Time, 1/1/2000.[1] The second step computes the obliquity of the ecliptic, $\epsilon = 23.439 - 0.0000004d$.[1] The third step computes the Mean of Date (MOD) lunar position. The Lunar position model is taken from the LanlGeoMag library[2]. The final step converts the result to a WGS84 rectangular-coordinate (x, y, z) unit vector (also by applying LanlGeoMag library routines).

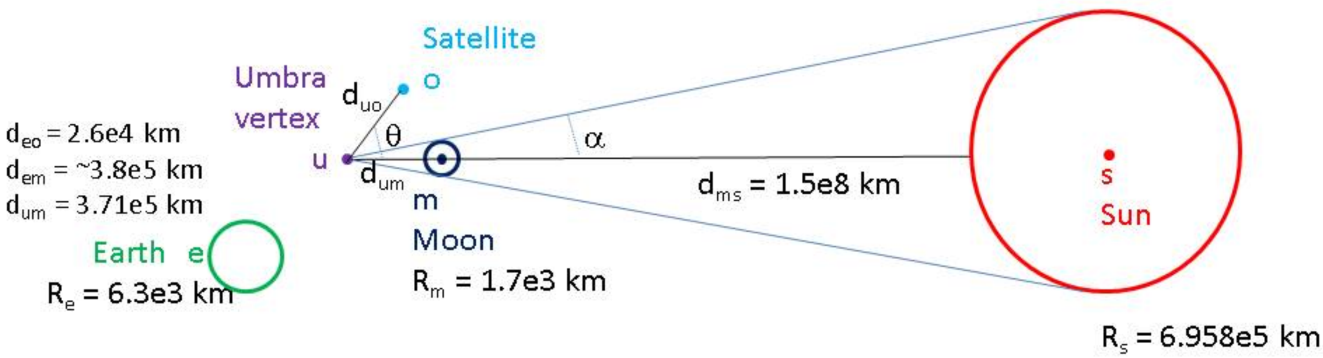
8 Determination if a Satellite lies within the Moon's Umbra

Figure 2 illustrates the geometry applied to compute whether a satellite lies within the shadow of the Moon. The Moon and Sun are both approximated to be spherical. Initial (known) required information includes the satellite position at a specified time. Then, the first two steps applies the routines described above to compute the positions of the Sun and Moon and their distance from the Earth. After subtracting the Earth-Moon vector from the Satellite and Sun positions, the problem reduces to the same mathematics as computing the Earth's umbra. Step 3 computes the umbra vertex position and angle α by applying the equations listed in Fig. 2. Step 4 computes the distance from the vertex to the satellite and the associated angle θ. Finally, if the vertex-to-satellite distance is less than the vertex-to-Moon distance and $\theta < \alpha$, then the satellite lies within the Moon's umbra.

The Earth's umbra distance is about $1.5 \times 10^6 km$. The Moon's umbra distance is about $3.71 \times 10^5 km$. The distance from the center of the Earth to the center of the Moon varies between $3.60 \times 10^5 km$ and $4.25 \times 10^5 km$, with a mean value of $3.84 \times 10^5 km$. A GPS satellite orbit radius is about $2.63 \times 10^4 km$. Thus, when the moon is farthest from Earth, a GPS satellite will not cross the Moon's umbra shadow. However, when the moon is closest to Earth, a GPS satellite, travelling othogonal to the umbra axis would cross the Moon's umbra in about 90 s. By comparison, a GPS satellite, travelling othogonal to the Earth's umbra axis would cross the Earth's umbra in about 55 min. The time crossing the Earth's Umbra was estimated assuming a non-moving umbra. Because the Earth spins, the umbra rotates the Earth once a day. Thus, if the satellite moves with or against the east-to-west umbra motion, umbra crossing times vary by a factor of from 2 to 0.5, respectively. The large time difference between the Earth and Moon umbra crossing times occurs because the GPS satellite is located so much closer to the Earth. Longer (up to about 21 minutes) satellite travel time within the Moon's umbra could be achieved when the Moon's umbra axis lies in the same plane and is nearly tangential to the satellite orbit.

The function, *IsInMoonUmbraShadow(t,***o***)*, returns a *true* if a satellite lies in the Moon's shadow. The input parameters specify the time, t, and the satellite ITRF orbital position, $\mathbf{o}$.

Figure 2. Does a Satellite lie within the Moon's Shadow ?



Algorithm Steps:

0. know **o** & time, t
1. Calc **s** & d_{es} [SunPositionFromEarth()]
2. Calc **m** , d_{em} & d_{ms}
3. Calc d_{um} , **u** , α [$d_{um} = d_{ms} R_m / (R_s - R_m)$; **u** = $- d_{ue}$ $\mathbf{s}_{ms} / d_{ms}$; $\sin\alpha = R_m / d_{um}$]
4. Calc d_{uo} & θ [**o.u** = ou $\cos\gamma$; $\sin\theta = d_{om} \sin\gamma / d_{um}$]
5. *If* ($d_{uo} < d_{um}$ and $\theta < \alpha$ *), in shadow*

Final Function: *tle::IsInMoonUmbraShadow(t,o)* Assumption: *Spherical Moon*

9 The Number of Minutes (per 12 hrs) a Satellite Spends within the Earth's Umbra

By summing for each minute over a 12-hr satellite orbit, it is straightforward to compute the total number of minutes spent within the Earth's Umbra. The functionality described in Section 6 is applied.

The function, *numberOfMinInUmbra(t,tle)*, returns the number of minutes per 12-hr orbit spent in the Earth's shadow. The input parameters specify the starting time, t, and the satellite orbit tle.

10 Conclusion

Two functions have been added to DIORAMA that compute the unit vector from either the center of the Earth to the Sun or from any observer's position to the Sun at some specified time. A third function computes if a satellite, at a given position and time, lies within the Earth's umbra (full shadow). Similarly, a function computes the unit vector from the Earth to the Moon. A fifth function computes if a satellite, at a given position and time, lies within the Moon's umbra (full shadow).

11 References

1. Wikipedia,"Position of the Sun", http://en.wikipedia.org/wiki/Position_of_the_Sun.

2. M. Henderson,"LANLGeoMag", http://www.rbsp-ect.lanl.gov.

3. J. Kouba,"A Simplified Yaw-Attitude Model for Eclipsing GPS Satellites",GPS Solut.(2009)13:1-12.