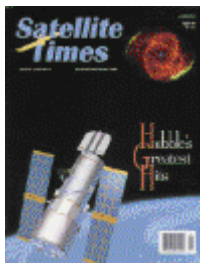
Orbital Coordinate Systems, Part II

By Dr. T.S. Kelso



November/December 1995

In our last column, we were working our way through the process of calculating the position and velocity of an observer on the Earth's surface in the Earth-Centered Inertial (ECI) reference frame. The goal, of course, was to be able to determine the position of the observer relative to an orbiting satellite to aid the process of visually acquiring or otherwise tracking the satellite from the ground.

At the end of the last column, we had seen that it would be impossible to determine an observer's position in the ECI reference frame without knowing that observer's local sidereal time, that is, the angle between the observer's meridian (longitude) and the vernal equinox. But because this time is measured relative to the stars and not the Sun, the equations to do the calculations are a bit complicated. Let's start of this column with a brief review of the equation for the local sidereal time and then explore, bit by bit, how to develop a computer routine to calculate this value for us. We will then continue the process by calculating the remaining information needed to determine an observer's position in the ECI reference frame.

As we saw last time, the local sidereal time, $\theta(\tau)$, can be calculated by adding the observer's ***east*** longitude, λ_E, to the Greenwich sidereal time (GST), $\theta_g(\tau)$. Using an equation from Page 50 of the ***Explanatory Supplement to the Astronomical Almanac***, we can first determine $\theta_g(0^h)$, the Greenwich sidereal time at 0^h (midnight) UTC, from which

$$\theta_g(\tau) = \theta_g(0^h) + \omega_e \cdot \Delta\tau$$

where $\Delta\tau$ is the UTC time of interest and $\omega_e = 7.29211510 \times 10^{-5}$ radians/second is the Earth's rotation rate. Recalling that $\theta_g(0^h)$ is given as

$$\theta_g(0^h) = 24110^s.54841 + 8640184^s.812866\ \boldsymbol{T_u} + 0^s.093104\ \boldsymbol{T_u}^2 - 6.2 \times 10^{-6}\ \boldsymbol{T_u}^3$$

where $\boldsymbol{T_u} = \boldsymbol{d_u}/36525$ and $\boldsymbol{d_u}$ is the number of days of Universal Time elapsed since JD 2451545.0 (2000 January 1, 12^h UT1), we can now set about determing what we know and what we need to calculate.

Let's develop a top-down algorithm to see what we need to do. Our goal in this portion of the algorithm is to determine the value of θ_g at a particular time, τ. To do this, we need values of $\theta_g(0^h)$, ω_e, and $\Delta\tau$. The value of ω_e is fixed and $\Delta\tau$ is the fraction of the day since 0^h UTC or $\Delta\tau =$ ***fraction***(τ) in days. Therefore, we still need to know $\boldsymbol{T_u}$, which depends upon $\boldsymbol{d_u}$. But $\boldsymbol{d_u}$ depends upon knowing the number of days of Universal Time elapsed since JD 2451545.0. So, if we know the Julian Date (JD) of τ, we have everything we need. Our top-down algorithm then looks something like this:

> Calculate: θ_g which depends on ($\theta_g(0^h)$, ω_e, $\Delta\tau$); ω_e constant; $\Delta\tau =$ ***fraction***(τ)
>
> Calculate: $\theta_g(0^h)$ which depends on ($\boldsymbol{T_u}$); $\boldsymbol{T_u} = \boldsymbol{d_u}/36525$
>
> Calculate: $\boldsymbol{d_u}$ which depends on (JD(τ))
>
> Calculate: JD(τ)

Therefore, our most basic calculation is the determination of the Julian Date. The Julian Date can be calculated from the equation in Section 12.95 (Page 606) ***[actually, this should be Section 12.92 (Page 604)]*** of the ***Explanatory Supplement to the Astronomical Almanac***, or using the approach on Page 61 of ***Astronomical Algorithms*** by Jean Meeus. This latter text is an excellent reference source for many relevant calculations in orbital mechanics and is highly recommended.

Using Meeus' approach, the Pascal code for calculating the Julian Date of January 0.0 of any year would be:

```
Function Julian_Date_of_Year(year : double) : double;
{ Calculate Julian Date of 0.0 Jan year }
  var
    A,B : longint;
  begin
  year := year - 1;
  A := Trunc(year/100);
  B := 2 - A + Trunc(A/4);
  Julian_Date_of_Year := Trunc(365.25 * year)
                       + Trunc(30.6001 * 14)
                       + 1720994.5 + B;
  end; {Function Julian_Date_of_Year}
```

To calculate the Julian Date of any calendar date, we simply combine the Julian Date of that year with the day of the year, where the day of the year can be calculated as:

```
Function DOY(yr,mo,dy : word) : word;
  const
    days : array [1..12] of word = (31,28,31,30,31,30,31,31,30,31,30,31);
  var
    i,day : word;
  begin
  day := 0;
  for i := 1 to mo-1 do
    day := day + days[i];
  day := day + dy;
  if ((yr mod 4) = 0) and
     (((yr mod 100) <> 0) or ((yr mod 400) = 0)) and
     (mo > 2) then
    day := day + 1;
  DOY := day;
  end; {Function DOY}
```

As an example, the Julian Date of 0^h UTC on 1995 October 01 would be written as:

```
JD := Julian_Date_of_Year(1995) + DOY(1995,10,1);
```

with a result of 2449991.5 = 2449717.5 + 274. Therefore, $\boldsymbol{d_u}$ equals -1553.5 days and $\boldsymbol{T_u}$ equals -1553.5/36525. From this value of $\boldsymbol{T_u}$, $\theta_g(0^h)$ can now be calculated to be -343378.2154 seconds. Since there are 86,400 seconds in a day, this is the same time (angle) as 2221.7846 seconds, so the equivalent GMST is 0^h 37^m $01^s.7846$ or an angle of 9.257 degrees.

Now, if the time of interest on 1995 October 01 was 9^h UTC, we would have to add $\omega_e \cdot \Delta\tau$ to the value of $\theta_g(0^h)$, where $\Delta\tau$ = 32,400 seconds (9 hours). Being careful to use the proper units, our new GMST is 9^h 38^m $30^s.4928$ or 144.627 degrees.

We can consolidate our calculation of the Greenwich Mean Sidereal Time into the following simple Pascal function where the input is the Julian Date of the time of interest (in our example of 9^h UTC on 1995 October 01, JD is 2449991.875) and the output is the GMST in radians. For our test case, this should be 2.524218 radians.

```
Function ThetaG_JD(jd : double) : double;
{ Reference:  The 1992 Astronomical Almanac, page B6. }
  var
    UT,TU,GMST : double;
  begin
  UT   := Frac(jd + 0.5);
  jd   := jd - UT;
  TU   := (jd - 2451545.0)/36525;
  GMST := 24110.54841 + TU * (8640184.812866 + TU * (0.093104 - TU * 6.2E-6));
  GMST := Modulus(GMST + 86400.0*1.00273790934*UT,86400.0);
  ThetaG_JD := twopi * GMST/86400.0;
  end; {Function ThetaG_JD}
```

Now, we can complete our calculation of the ECI position of our observer. We'll start by assuming a spherical Earth and then go back and rework our solution, in our next column, using an oblate Earth. If our observer is located at 40° North latitude and 75° West longitude (near Philadelphia), we can easily calculate the $\boldsymbol{z}$ coordinate according to

$$\boldsymbol{z} = \boldsymbol{R_e} \sin \varphi$$

where $\boldsymbol{R_e}$ = 6378.135 km and φ = 40°. To calculate $\boldsymbol{x}$ and $\boldsymbol{y}$, we use

$$\boldsymbol{x} = \boldsymbol{R} \cos \theta$$

$$\boldsymbol{y} = \boldsymbol{R} \sin \theta$$

where

$$\theta = \theta_g + \lambda_E$$

$$\boldsymbol{R} = \boldsymbol{R_e} \cos \varphi.$$

Using our example, λ_E = -75° and θ_g = 144°.627, so θ = 69°.627 (remember, this is the local sidereal time—the angle between the observer's meridian and the vernal equinox) and $\boldsymbol{R}$ = 4885.935 km. Therefore, the ECI position of our observer at the time of interest is:

$$\boldsymbol{x} = 1700.938 \text{ km}, \boldsymbol{y} = 4580.302 \text{ km}, \boldsymbol{z} = 4099.786 \text{ km}.$$

After all that work calculating the local sidereal time, the rest of the calculation was relatively easy! We can encapsulate this calculation using the following simple Pascal procedure:

```
Procedure Calculate_User_Pos(lat,lon,alt,time : double;
                                  var x,y,z : double);
{ Reference:  The 1992 Astronomical Almanac, page K11. }
  const
    re = 6378.135;
  var
    theta : double;
  begin
  theta := Modulus(ThetaG_JD(time) + lon,twopi);
  r := (re + alt)*Cos(lat);
  x := r*Cos(theta);
  y := r*Sin(theta);
  z := (re + alt)*Sin(lat);
  end; {Procedure Calculate_User_Pos}
```

In the inputs for this routine, `lat` and `lon` are in radians, `alt` is in kilometers, and `time` is the Julian Date of interest. The outputs (`x`, `y`, and `z`) are in kilometers.

So, how do we now calculate the look angle to a satellite? If the satellite's position in the ECI coordinate system is defined as [x_S, y_S, z_S] and the observer is [x_O, y_O, z_O], then the range vector is simply

$$[r_x, r_y, r_z] = [x_S - x_O, y_S - y_O, z_S - z_O].$$

This vector, however, is in the ECI system and to generate look angles, we need it to be in the topocentric-horizon system shown in Figure 1. That system has its ***z*** axis pointing toward the zenith, the ***x*** axis pointing South, and the ***y*** axis pointing East.

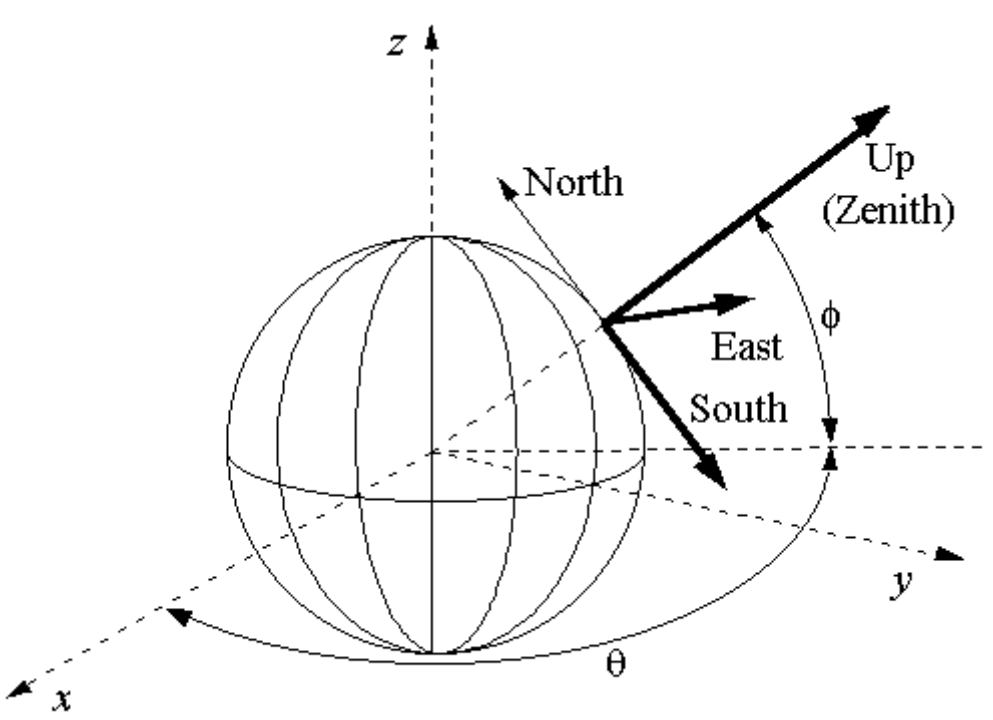


Figure 1. Topocentric-Horizon Coordinate System

To transform to the topocentric-horizon system, we must first rotate through an angle θ (the local sidereal time) about the ***z*** axis (Earth rotation axis) and then through an angle φ (the observer's latitude) about the ***y*** axis. The coordinates (r_S, r_E, r_Z) become:

$$r_S = \sin\varphi \cos\theta\, r_x + \sin\varphi \sin\theta\, r_y - \cos\varphi\, r_z$$

$$r_E = -\sin\theta\, r_x + \cos\theta\, r_y$$

$$r_Z = \cos\varphi \cos\theta\, r_x + \cos\varphi \sin\theta\, r_y + \sin\varphi\, r_z$$

The range to the satellite is

$$r = \sqrt{[r_S^2 + r_E^2 + r_Z^2]},$$

the elevation is

$$\mathrm{El} = \sin^{-1}(r_Z / r),$$

and the azimuth is

$$\mathrm{Az} = \tan^{-1}(-r_E / r_S).$$

The minus sign is necessary because azimuth is measured clockwise from North instead of counter-clockwise from South (which would be standard for a right-handed orthogonal coordinate system). Care must be taken with the azimuth to ensure the proper quadrant is selected for the arctangent.

We can calculate the look angles using the procedure described with the Pascal procedure `Calculate_Look`, shown below. The inputs are the satellite's ECI coordinates (`xs`, `ys`, and `zs`) in kilometers, the observer's latitude and longitude (`lat` and `lon`) in radians and altitude (`alt`) in kilometers, along with the time of interest (`time`) as a Julian Date. The outputs are the azimuth and elevation (`az` and `el`) in radians and the range (`rg`) in kilometers.

```
Procedure Calculate_Look(xs,ys,zs,lat,lon,alt,time : double;
                                    var az,el,rg : double);
  var
    xo,yo,zo,
    rx,ry,rz,
    theta,
    top_s,top_e,top_z : double;
  begin
  Calculate_User_Pos(lat,lon,alt,time,xo,yo,zo);
  theta := Modulus(ThetaG_JD(time) + lon,twopi);
  rx := xs - xo;
  ry := ys - yo;
  rz := zs - zo;
  top_s := Sin(lat)* Cos(theta)*rx
         + Sin(lat)* Sin(theta)*ry
         - Cos(lat)*rz
  top_e := - Sin(theta)*rx
         + Cos(theta)*ry;
  top_z := Cos(lat)* Cos(theta)*rx
         + Cos(lat)* Sin(theta)*ry
         + Sin(lat)*rz;
  az := ArcTan(-top_e/top_s);
  if top_s > 0 then
    az := az + pi;
  if az < 0 then
    az := az + twopi;
  rg := Sqrt(rx*rx + ry*ry + rz*rz);
  el := ArcSin(top_z/rg);
  end; {Procedure Calculate_Look}
```

To sum things up, in order to calculate the look angles for a satellite relative to an observer on the ground, we must first calculate the satellite's position in the ECI coordinate system, then calculate the observer's ECI position, take the difference of the two vectors, and then transform (rotate) the vector from the ECI coordinate frame to the topocentric-horizon frame. The most difficult part of this process is in calculating the Earth's rotation angle when determining the observer's position.

As always, if you have questions or comments on this column, feel free to send me e-mail at TS.Kelso@celestrak.com or write care of ***Satellite Times***. Until next time, keep looking up!

Dr. T.S. Kelso [TS.Kelso@celestrak.com]
Follow CelesTrak on Twitter @TSKelso
Last updated: 2019 Dec 27 03:29:00 UTC
Accessed 129,349 times
Current system time: 2020 Jun 26 12:59:05 UTC
CelesTrak's Simple Privacy Policy