Computing satellite velocity using the broadcast ephemeris

Benjamin W. Remondi

Introduction

GPS has been used for applications where satellite velocities are needed as well as satellite positions. Traditionally, the position was determined from noisy code pseudoranges while the velocity came from delta carrier phase measurements. This motivates the need to have a velocity model to accompany the velocity data. With velocity data, the velocity states are directly observed in a Kalman filter. While processing can be carried out using either post-processed or predicted ephemerides from the international GPS service (IGS), it is convenient to use broadcast ephemerides for real-time applications. In this paper, a set of equations is derived which will allow one to compute the velocity vector for a GPS satellite using time derivatives of the Keplerian elements and correction terms broadcast from the GPS satellites, as described in the GPS interface control document (ICD-GPS-200). It will be assumed that the reader is familiar with the equations listed in this document for computing the X, Y, Z earth-centered, earth-fixed (ECEF) coordinates of a satellite (the reader may obtain a PDF version of the ICD-GPS-200 from the US Coast Guard website at http://www.navcen,uscg.gov/pubs/gps/icd200/default.htm). A complete main program written in C is included on the GPS Toolbox website (http://www.ngs.noaa.gov/gps-toolbox) to illustrate how these equations can be easily incorporated into source code to yield accurate velocity components. As a check, the velocity calculations were done numerically as well.

Received: 19 March 2004 / Accepted: 26 March 2004
Published online: 4 August 2004

B. W. Remondi (✉)
The XYZs, of GPS, Inc., P.O. Box 37, Dickerson, MD 20842, USA
E-mail: remondi@xyzsofgps.com
Tel.: +1-301-9727402
Fax: +1-301-3492547

Derivation of the velocity equations

Let us first list the symbols needed to compute the broadcast orbits. These are listed below using the notation found in the ICD-GPS-200.

$$c_{\mathrm{rs}}, \Delta n, M_0, c_{\mathrm{uc}}, e, c_{\mathrm{us}}, (A)^{1/2}, t_{\mathrm{oe}}, c_{\mathrm{ic}}, (OMEGA)_0,$$
$$c_{\mathrm{is}}, i_0, c_{\mathrm{rc}}, \omega, OMEGADOT, IDOT$$

These parameters are broadcast from the GPS satellites in units of meters, radians, seconds, semi-circles, and semi-circles/second. One semi-circle is the same as 180°. In what follows, we will assume that any values sent in semi-circles or semi-circles/second have been converted to radians or radians/second, respectively.
There are also some needed constants given as follows.

$$\text{Earth rotation rate}: \Omega_e = 7.2921151467 \times 10^{-5}\,\mathrm{rad/s}$$

WGS 84 value for the product of earth's gravity and its mass:

$$\mu = 3.986005 \times 10^{14}\,\mathrm{m^3/s^2}$$

Next let us rewrite the broadcast computation algorithm, to compute ECEF position at user time t, from the ICD-GPS-200, adding comments and new steps for clarity and for derivatives. Please refer to the ICD-GPS-200 in what follows. In the discussion below, the new equations and comments added for computing velocity will be preceded by two slashes (i.e., those lines either not found in the ICD-GPS-200 document or not needed for computing position begin with the characters '//').

$A = (A^{1/2})^2$
$n_0 = \sqrt{\mu/A^3}$
$t_k = t - t_{\mathrm{oe}}$ t is the time of the desired orbit position
$n = n_0 + \Delta n$
$M_k = M_0 + n t_k$
$//\dot{M}_k = n$ Consider all lines beginning with //
to be a comments.
$//M_k = E_k - e \sin E_k$
$E_k = M_k$ // Initialize eccentric anomaly to mean anomaly
for $(i = 0;\ i<7;\ i++)\ E_k = M_k + e \sin E_k$ //
Improve E_k; more loops than needed
$//\dot{M}_k = \dot{E}_k - e \cos E_k \dot{E}_k = (1 - e \cos E_k)\dot{E}_k$
$//\dot{E}_k = \dot{M}_k/(1 - e \cos E_k)$ will be needed later

//True anomaly is υ_k

$$//\upsilon_k = \tan^{-1}\left\{\frac{\sin\upsilon_k}{\cos\upsilon_k}\right\} = \tan^{-1}\left\{\frac{\sqrt{1-e^2}\sin E_k(1-e\cos E_k)}{(\cos E_k - e)/(1-e\cos E_k)}\right\}$$

$$// = \tan^{-1}\left\{\frac{\sqrt{1-e^2}\sin E_k}{(\cos E_k - e)}\right\}$$

//In practice the above equation is optional (see below), but we will
//use it because the ATAN2 function takes care of the ambiguity nicely.

//Because we know E_k the following (ambiguous) equation could
//have been used for υ_k as well. $E_k = \cos^{-1}\left\{\frac{e+\cos\upsilon_k}{1+e\cos\upsilon_k}\right\}$

//This equation instead will be used to compute $\dot{\upsilon}_k$.

$$// \cos E_k(1+e\cos\upsilon_k) = e + \cos\upsilon_k$$

//Differentiating both sides we get

$$// -\sin E_k\dot{E}_k(1+e\cos\upsilon_k) - \cos E_k e\sin\upsilon_k\dot{\upsilon}_k = -\sin\upsilon_k\dot{\upsilon}_k$$

$$// -\sin E_k\dot{E}_k(1+e\cos\upsilon_k) = (-1+\cos E_k e)\sin\upsilon_k\dot{\upsilon}_k$$

$$//\dot{\upsilon}_k = \sin E_k\dot{E}_k(1+e\cos\upsilon_k)/[(1-\cos E_k e)\sin\upsilon_k]$$

$$\Phi_k = \upsilon_k + \omega$$

$$//\dot{\Phi}_k = \dot{\upsilon}_k \quad //\omega \text{ is assumed to be constant}$$

$$\delta u_k = c_{us}\sin(2\Phi_k) + c_{uc}\cos(2\Phi_k)$$

$$\delta r_k = c_{rs}\sin(2\Phi_k) + c_{rc}\cos(2\Phi_k)$$

$$\delta i_k = c_{is}\sin(2\Phi_k) + c_{ic}\cos(2\Phi_k)$$

$$//\dot{\delta}u_k = 2[c_{us}\cos(2\Phi_k) - c_{uc}\sin(2\Phi_k)]\dot{\Phi}_k$$

$$//\dot{\delta}r_k = 2[c_{rs}\cos(2\Phi_k) - c_{rc}\sin(2\Phi_k)]\dot{\Phi}_k$$

$$//\dot{\delta}i_k = 2[c_{is}\cos(2\Phi_k) - c_{ic}\sin(2\Phi_k)]\dot{\Phi}_k$$

$$u_k = \Phi_k + \delta u_k \quad //\text{Same as } u_k = \upsilon_k + \omega + \delta u_k$$

$$r_k = A(1-e\cos E_k) + \delta r_k$$

$$i_k = i_0 + (IDOT)t_k + \delta i_k$$

$$//\dot{u}_k = \dot{\Phi}_k + \dot{\delta}u_k \quad //\text{Same as } \dot{u}_k = \dot{\upsilon}_k + \dot{\delta}u_k$$

$$//\dot{r}_k = Ae\sin E_k\dot{E}_k + \dot{\delta}r_k$$

$$\dot{i}_k = IDOT + \dot{\delta}i_k$$

$$x'_k = r_k\cos u_k$$

$$y'_k = r_k\sin u_k$$

$$//\dot{x'}_k = \dot{r}_k\cos u_k - r_k\sin u_k\dot{u}_k$$

$$//\dot{x'}_k = \dot{r}_k\cos u_k - y'_k\dot{u}_k$$

$$//\dot{y'}_k = \dot{r}_k\sin u_k + r_k\cos u_k\dot{u}_k$$

$$//\dot{y'}_k = \dot{r}_k\sin u_k + x'_k\dot{u}_k$$

$$\Omega_k = \Omega_0 + \left(\dot{\Omega} - \dot{\Omega}_e\right)t_k - \dot{\Omega}_e t_{oe}$$

$$//\dot{\Omega}_k = \left(\dot{\Omega} - \dot{\Omega}_e\right)$$

$$x_k = x'_k\cos\Omega_k - y'_k\cos i_k\sin\Omega_k$$

$$y_k = x'_k\sin\Omega_k + y'_k\cos i_k\cos\Omega_k$$

$$z_k = y'_k\sin i_k$$

$$//\dot{x}_k = \dot{x}'_k \cos\Omega_k - x'_k \sin\Omega_k \dot{\Omega}_k - \dot{y}'_k \cos i_k \sin\Omega_k - y'_k \left(-\sin i_k \dot{i}_k \sin\Omega_k + \cos i_k \cos\Omega_k \dot{\Omega}_k\right)$$
$$//\dot{x}_k = \dot{x}'_k \cos\Omega_k - \dot{y}'_k \cos i_k \sin\Omega_k + y'_k \sin i_k \sin\Omega_k \dot{i}_k - \left(x'_k \sin\Omega_k + y'_k \cos i_k \cos\Omega_k\right)\dot{\Omega}_k$$
$$//\dot{x}_k = \dot{x}'_k \cos\Omega_k - \dot{y}'_k \cos i_k \sin\Omega_k + y'_k \sin i_k \sin\Omega_k \dot{i}_k - y_k \dot{\Omega}_k$$
$$//\dot{y}_k = \dot{x}'_k \sin\Omega_k + x'_k \cos\Omega_k \dot{\Omega}_k + \dot{y}'_k \cos i_k \cos\Omega_k + y'_k \left(-\sin i_k \dot{i}_k \cos\Omega_k - \cos i_k \sin\Omega_k \dot{\Omega}_k\right)$$
$$//\dot{y}_k = \dot{x}'_{//k} \sin\Omega_k + \dot{y}'_k \cos i_k \cos\Omega_k - y'_k \sin i_k \dot{i}_k \cos\Omega_k + \left(x'_k \cos\Omega_k - y'_k \cos i_k \sin\Omega_k\right)\dot{\Omega}_k$$
$$//\dot{y}_k = \dot{x}'_k \sin\Omega_k + \dot{y}'_k \cos i_k \cos\Omega_k - y'_k \sin i_k \dot{i}_k \cos\Omega_k + x_k \dot{\Omega}_k$$
$$//\dot{z}_k = \dot{y}'_k \sin i_k + y'_k \cos i_k \dot{i}_k$$

Example source code

The GPS Toolbox website (http://www.ngs.noaa.gov/gps-toolbox) lists the source code for an example C program, which illustrates how to compute both position and velocity for PRN 20 starting with broadcast ephemeris data similar to what might be found in a typical RINEX navigation message file. The reader can use this C code to create a similar subroutine or function, which will compute the position and velocity of a satellite given the broadcast ephemeris data, the PRN name, and a requested GPS time (transmission time) in units of GPS week and seconds-of-week. The author has intentionally presented this in a "main" program so as not to obscure the tutorial. In practice, a pointer to a broadcast structure would be passed to a similar C function. It should be pointed out that the ambiguity with respect to the GPS week has been ignored in this simple tutorial program.

Additional remarks

The above mentioned C program was rerun at both t+0.005 s and at t−0.005 s to determine the position of the satellite so as to determine the velocity numerically using delta position divided by delta time (i.e., over 0.01 s). This relatively large delta time was used in recognition of the finite precision of the computer. This is why a "long double" (80 bit) was adopted in the source code. In practice, a long double is not required and a "double" (64 bit) will suffice. If a small delta t (e.g., 1 μs) were used, this simple numerical approach would have been very poor. If a larger delta t were used (e.g., 10 s), the simple derivative definition would not suffice and a polynomial fit over a number of points would be needed.

The numerical and analytical results agree sufficiently to ensure that the analytical (symbolic) derivatives were done correctly—and that is the point. Separately, the velocity was compared with the precise orbits and the agreement was better than 1 mm/s, however this is not the point of this article.

Summary and conclusions

In summary, this paper has presented an expansion of the ICD-GPS-200 algorithm to include computation of the velocity components of the satellite in the ECEF frame. It has been verified that the numerical and analytical computations agree within about 1 μ/s. This is just what one would expect from a "long double" comparison. In a future ICD algorithm, possibly the algorithm could be expanded to include velocity. With proper care, the velocity components can be computed numerically as well.