GPS EASY Suite II

A Matlab Companion

KAI BORRE
AALBORG UNIVERSITY

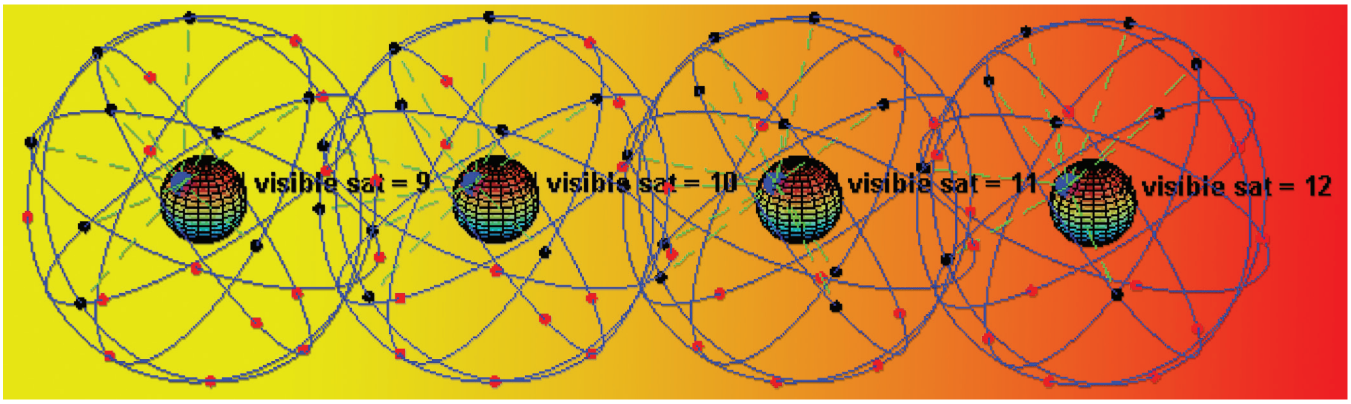


Image of GPS constellation based on public domain file from Wikimedia Commons

Many practitioners in the GNSS field are familiar with Matlab, a high-level technical computing language and interactive environment for algorithm development, data visualization, data analysis, and numeric computation. In this new series, a prominent Danish GNSS researcher uses Matlab to illustrate and explain a variety of common GPS issues.

In 2003 I published a paper called "The GPS EASY Suite — Matlab Code for the GPS Newcomer." The paper consisted of 10 parts, which are cited in the accompanying sidebar, "EASY Suite I Topics." Each installment included a printed text and a file of related Matlab code or scripts (an *M*-file) that could be downloaded from a designated web site.

Each part contained an answer to often-asked questions in my classes. I did not, however, only offer an answer to the questions, but also a Matlab code that solves the problem. The article became a tremendous success and remained the most downloaded paper from the *GPS Solutions* site for more than a year.

Now, students love this sort of support. But I also received positive reactions from professionals who used the code in research papers, which saved them a lot of coding efforts.

The original Matlab code also turned out to be the most downloaded file from the Aalborg website. It resulted in numerous e-mails from interested readers asking for more files. These requests now answered by the creation of eight additional *M*-files. Some involve more complex problems and coding.

Since 2003, I have received a steady flow of additional questions from readers of Gilbert Strang's and my book, *Linear Algebra, Geodesy, and GPS*. Last summer I decided to create another set of common GPS problems/solutions and accompanying Matlab code — EASY Suite II. These will be presented in serial fashion, as one or more exercises in this and forthcoming issues of *Inside GNSS*.

Introducing the New Suite

Easy Suite II augments the set of basic computational tasks with the following topics:

EASY11, stereographic sky plot of satellite orbits and plot of time when satellites are above a given local horizon

EASY12, details of the LAMBDA method, explained through a small numerical example

EASY13, receiver autonomous integrity monitoring (RAIM), horizontal protection level (HPL), and vertical protection level (VPL)

EASY14, sample of space-based augmentation system (SBAS) corrected positions and their presentation in Stanford plots

EASY15, accuracy comparison between pseudorange based stand-alone positions, baselines computed using pseudoranges alone, and pseudoranges and carrier phase observations

EASY16, error analysis of a selected one-way observation

EASY17, satellite orbits in inertial and Earth-centered, Earth-fixed (ECEF) systems, and curve defined by sub-satellite points

EASY18, computation of differential corrections at a base station.

The original suite was thought of as comprising the base for an elementary course in GPS while, as reflected in the subjects, the new suite is of a more optional, topical character.

In 2003 and later, many users encountered difficulty in finding the necessary files for running individual EASY-files. Therefore, I chose to copy all files needed into single directories accessible online, so that each directory becomes self-contained. The price we pay is multiple copies of basic *M*-files.

The complete set of Easy Suite II Matlab codes can be found in compressed ("zipped") files at <http://gps.aau.dk/~borre/easy2>. Readers can find much of the theoretical background for the EASY scripts in Chapters 14 and 15 of *Linear Algebra, Geodesy, and GPS* (see Additional Resources for details). However, we have added some text that emphasizes certain issues that are central in the codes, but may be difficult to find in a textbook.

EASY11: GPS Sky Plots

The EASY-Suite starts with a polar plot of satellite orbits (see Figure 1) as viewed from a given location, a graph showing the number of visible satellites, and the period of time when they are visible during 24 consecutive hours.

EASY11 itself is based on an almanac downloaded most easily from the National Geodetic Survey (NGS) website <http://www.ngs.noaa.gov/CORS/Data.html>. The actual file name is brdc1550.08n.

The RINEX file has been reformatted into Matlab's binary format for satellite ephemerides using the *M*-file rinexe. The user enters (φ,λ) for the position on the ground where the plot is to be used, and a value for the elevation mask to be set. Then the azimuth and elevation angles for all visible positions of the included satellites are computed and plotted as polar coordinates.

Next follows bookkeeping on how many and which satellites are visible during the day and the time periods when they can be seen. Figures 2 and 3 show the resulting plots.

The Matlab code is simple, the result is impressive, and useful. In early GPS days when the constellation was incomplete, such plots were especially valuable for planning purposes to be sure that enough satellites would be available for positioning. Except in the most severe terrain and urban canyons, receivers can find plenty of GPS (and GLONASS) satellites around the clock, and this situation will become even more pronounced when Galileo satellites are launched.

EASY12: LAMDA Method

Most students find the theory behind the Least-squares AMBiguity Decorrelation Adjustment (LAMBDA) for ambiguity resolution difficult to understand. We shall try to smooth the path.

First we describe in a subsection the linear algebra involved. Next we add an M-script that elucidates how the method works on a simple case with three ambiguities.

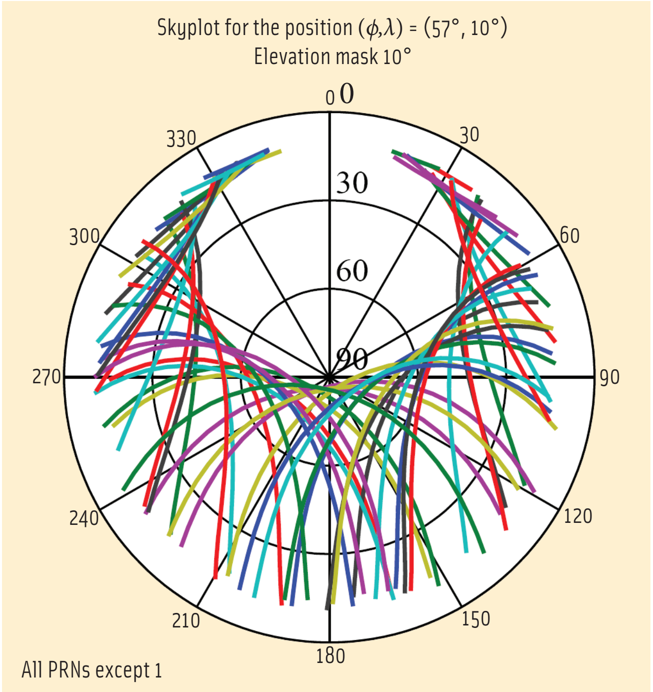


FIGURE 1 Sky plot including all GPS satellites for a period of 24 hours at a given position. The elevation mask is 10 degrees.

EASY Suite I Topics

The original Easy Suite, which can be found online at <http://kom.aau.dk/~borre/easy/>, dealt with the following issues:

- EASY1, time conversion
- EASY2, computation of a satellite's position from an ephemeris
- EASY3 computation of a receiver's position from pseudoranges
- EASY4 computation of a baseline from pseudoranges alone
- EASY5 computation of a baseline from pseudoranges and phase observations using a least-squares solution
- EASY6 the same, but now using a Kalman filter for the baseline estimation
- EASY6e the same, but introducing down-weighting of older observations
- EASY7 estimation of receiver clock offset
- EASY8 check of cycle slips and receiver clock reset
- EASY9 various coordinate representations of a given baseline
- EASY10 estimation of ionospheric delay for the individual satellites

Linear Algebra for LAMBDA

Let the vector $\boldsymbol{b}$ contain the three components of the baseline and the vector $\boldsymbol{a}$ contain ambiguities for the L_1 frequency and possibly for the L_2 frequency. The d*ouble differenced observations* are collected in the vector $\boldsymbol{y}$:

$$\begin{bmatrix} B & A \end{bmatrix} \begin{bmatrix} \boldsymbol{b} \\ \boldsymbol{a} \end{bmatrix} = \boldsymbol{y} + errors. \quad (1)$$

The normal equations are

$$\begin{bmatrix} B^T B & B^T A \\ A^T B & A^T A \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{b}} \\ \hat{\boldsymbol{a}} \end{bmatrix} = \begin{bmatrix} B^T \\ A^T \end{bmatrix} \boldsymbol{y} \quad (2)$$

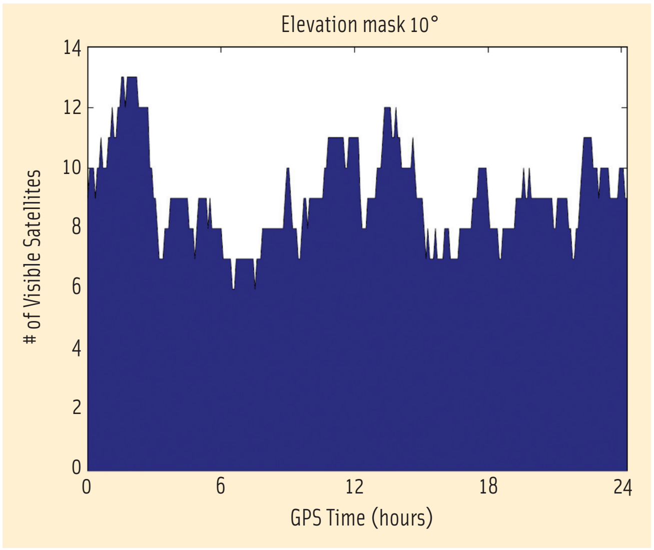


FIGURE 2 Number of visible satellites

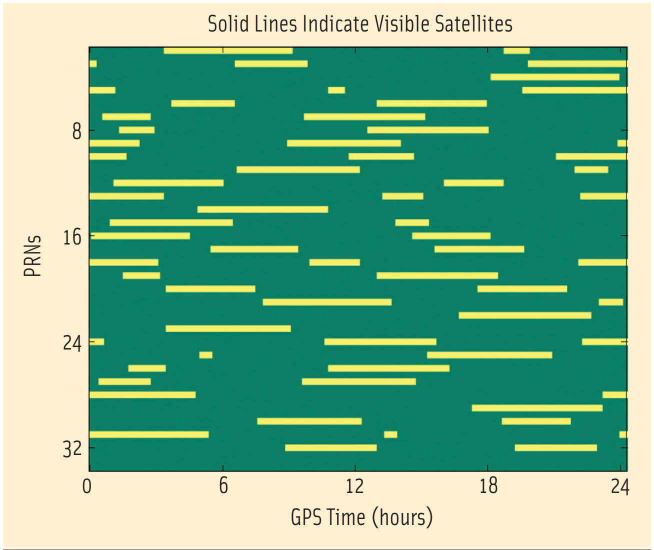


FIGURE 3 Time period when the visible satellites are 10 degrees or higher above the horizon

and the formal solution is

$$\begin{bmatrix} \hat{\boldsymbol{b}} \\ \hat{\boldsymbol{a}} \end{bmatrix} = \begin{bmatrix} B^{\mathrm{T}}B & B^{\mathrm{T}}A \\ A^{\mathrm{T}}B & A^{\mathrm{T}}A \end{bmatrix}^{-1} \begin{bmatrix} B^{\mathrm{T}} \\ A^{\mathrm{T}} \end{bmatrix} \boldsymbol{y} = \begin{bmatrix} Q_{\hat{b}} & Q_{\hat{b}\hat{a}} \\ Q^{T}_{\hat{b}\hat{a}} & Q_{\hat{a}} \end{bmatrix} \begin{bmatrix} B^{\mathrm{T}} \\ A^{\mathrm{T}} \end{bmatrix} \boldsymbol{y}. \tag{3}$$

The components of the solution vector $\hat{\boldsymbol{a}}$ are reals; however, we want a solution $\check{\boldsymbol{a}}$ of *integers*! Hence, we find an integer vector $\boldsymbol{a}$ such that

$$(\hat{\boldsymbol{a}} - \boldsymbol{a})^{\mathrm{T}} Q^{-1}_{\hat{a}} (\hat{\boldsymbol{a}} - \boldsymbol{a}) = \textit{min. over integer vectors } \boldsymbol{a}. \tag{4}$$

After the integer solution $\check{\boldsymbol{a}}$ is found we substitute it for $\hat{\boldsymbol{a}}$. Consequently, the solution for $\hat{\boldsymbol{b}}$ changes to $\check{\boldsymbol{b}}$. In order to determine $\check{\boldsymbol{b}}$, we multiply the lower block row in (3) by $Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}}$ and subtract from the upper block row:

$$\begin{bmatrix} \hat{\boldsymbol{b}} - Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}}\,\hat{\boldsymbol{a}} \\ \hat{\boldsymbol{a}} \end{bmatrix} = \begin{bmatrix} Q_{\hat{b}} - Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}} & 0 \\ Q^{\mathrm{T}}_{\hat{b}\hat{a}} & Q_{\hat{a}} \end{bmatrix} \begin{bmatrix} B^{\mathrm{T}} \\ A^{\mathrm{T}} \end{bmatrix} \boldsymbol{y}. \tag{5}$$

The upper block row gives:

$$\hat{\boldsymbol{b}} - Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}}\,\hat{\boldsymbol{a}} = \left(Q_{\hat{b}} - Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}}\right)B^{\mathrm{T}}\boldsymbol{y}.$$

The right side is known and constant. If we change $\hat{\boldsymbol{a}}$ to $\check{\boldsymbol{a}}$, then $\hat{\boldsymbol{b}}$ changes to $\check{\boldsymbol{b}}$ and we have:

$$\hat{\boldsymbol{b}} - Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}}\,\hat{\boldsymbol{a}} = \check{\boldsymbol{b}} - Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}}\,\check{\boldsymbol{a}}$$

or

$$\check{\boldsymbol{b}} = \hat{\boldsymbol{b}} - Q_{\hat{b}\hat{a}}Q^{-1}_{\hat{a}}\left(\hat{\boldsymbol{a}} - \check{\boldsymbol{a}}\right). \tag{6}$$

The right side is known and $\check{\boldsymbol{b}}$ is quickly found.

Minimizing a Quadratic Expression over Integer

The LAMBDA method starts as other least-squares computations by solving the normal equations (3). The result is partly the vector of real-valued ambiguities $\hat{\boldsymbol{a}}$ and partly the pertinent covariance matrix $Q_{\hat{a}}$.

The core problem can be stated in a very clear form

$$\textit{minimize } (\boldsymbol{I} - \hat{\boldsymbol{I}})^{\mathrm{T}} Q^{-1}_{\hat{I}} (\boldsymbol{I} - \hat{\boldsymbol{I}}) \quad \textit{over integer vector } \boldsymbol{I}.$$

In order to emphasize the integer nature of the problem we have swapped the notation from $\boldsymbol{a}$ to $\boldsymbol{I}$.

This is the problem we study, and in one case this is especially simple. If $Q^{-1}_{\hat{I}}$ is diagonal, the best vector comes from rounding each component of $\hat{\boldsymbol{I}}$ to the nearest integer.

The components are uncoupled when $Q^{-1}_{\hat{I}}$ is diagonal. The quadratic is purely a sum of squares $\Sigma(Q^{-1}_{\hat{I}})_{jj}\ (\boldsymbol{I}_j - \hat{\boldsymbol{I}}_j)^2$. The minimum comes by making each term as small as possible. So the best $\check{\boldsymbol{I}}_j$ is the integer nearest to $\hat{\boldsymbol{I}}_j$.

In practice the individual $\hat{\boldsymbol{I}}_j$ components are highly correlated. Recalling that we often deal with 20 ambiguities, we realize that a search procedure to find the minimum is unrealistic. Therefore, an idea about decorrelating the ambiguities as much as possible, before starting the search, should lead to an effective procedure.

This is just what the LAMBDA method does. It was described by Peter Teunissen in 1993 and is, from a theoretical point of view, still considered to be the best method.

The first step consists in transforming $Q^{-1}_{\hat{I}}$ such that its off-diagonal entries become numerically smaller. These entries measure the correlation.

We start by computing the LDL^{T} decomposition of the given covariance matrix

$$Q_{\hat{I}} = LDL^{T}.$$

Next we construct a matrix of integers Z from L by a sequence of *integer Gauss transformations* and *permutations* such that

$$Q_{\bar{I}} = Z^{T}Q_{\hat{I}}Z$$

is as "diagonal" as possible. Now the search defined by (4) is substituted by a search over integers $\boldsymbol{I}$:

$$(\hat{\boldsymbol{I}}-\boldsymbol{I})^{\mathrm{T}} Z^{\mathrm{T}} Q_{\hat{I}}^{-1} Z(\hat{\boldsymbol{I}}-\boldsymbol{I}) = min. \quad (7)$$

The choice of Z is based on integer elimination starting with the first row of L. This will certainly involve row exchanges and, therefore, Z will not be triangular. The essential idea for this *lattice basis reduction* was given by A. K. Lenstra et alia in 1982 (See Additional Resources), and the algorithm is sometimes called L^3. For further details see Strang and Borre, pages 495–499.

The result of the search is the integer vector $\check{\boldsymbol{I}}$. Finally, $\check{\boldsymbol{I}}$ is transformed back to the ambiguity space according to

$$\bar{\boldsymbol{I}} = (Z^{-1})^{\mathrm{T}} \check{\boldsymbol{I}}. \quad (8)$$

Note that both Z and Z^{-1} must have integer entries. A consequence of this is that det(Z) = 1 as seen from Cramer's rule. The condition on the determinant of Z means that the Z transformation preserves the search volume. At the end of this process, we have transformed a highly correlated space (elongated ellipses) into a sphere-like search space, which diminishes the search time tremendously.

EASY12 illustrates the computational steps including a few numerical details. We choose a numerical example from P. J. de Jonge and C. C. J. M. Tiberius (1996). Further computational details are well described in that report.

A Numerical Example

Given the float ambiguities

$$\hat{\boldsymbol{I}} = \begin{bmatrix} 5.45 \\ 3.1 \\ 2.97 \end{bmatrix} \quad (9)$$

with covariance matrix

$$Q_{\hat{I}} = \begin{bmatrix} 6.290 & 5.978 & 0.544 \\ 5.978 & 6.292 & 2.340 \\ 0.544 & 2.340 & 6.288 \end{bmatrix}. \quad (10)$$

We introduce integer shifts of $\hat{\boldsymbol{I}}$ in order to secure that $-1 < \hat{\boldsymbol{I}}_j \le 1$:

$$\begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix}$$

This leaves the remainders

$$\hat{\boldsymbol{I}}_r = \begin{bmatrix} 0.45 \\ 0.10 \\ 0.97 \end{bmatrix}.$$

Now we factorize $Q_{\hat{I}}$ into LDL^{T} with the lower triangular matrix

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1.065 & 1 & 0 \\ 0.087 & 0.372 & 1 \end{bmatrix} \quad (11)$$

and the diagonal matrix

$$D = \begin{bmatrix} 0.090 & 0 & 0 \\ 0 & 5.421 & 0 \\ 0 & 0 & 6.288 \end{bmatrix}. \quad (12)$$

We then compute the initial size of the search ellipsoid as the squared distances of partially rounded float vectors to the float vector in the metric of the covariance matrix. In the present example the search volume is determined by $\chi^2 = 1.245$.

The call $[Q_{\bar{I}}, Z, L_t, D_t] = \mathrm{decorrel}(Q_{\hat{I}}, \hat{\boldsymbol{I}}_r)$ computes the integer transformation matrix Z, a decorrelated covariance matrix $Q_{\bar{I}}$, and the transformed version of the LDL^{T} decomposition. All quantities are mentioned below:

$$Z = \begin{bmatrix} -2 & 3 & 1 \\ 3 & -3 & -1 \\ -1 & 1 & 0 \end{bmatrix}. \quad (13)$$

The transformed and shifted ambiguities are

$$\boldsymbol{I}_s = Z^{\mathrm{T}} \hat{\boldsymbol{I}}_r = \begin{bmatrix} -1.57 \\ 2.02 \\ 0.35 \end{bmatrix}. \quad (14)$$

The transformed, decorrelated covariance matrix is

$$Q_{\bar{I}} = Z^{\mathrm{T}} Q_{\hat{I}} Z = \begin{bmatrix} 4.476 & 0.334 & 0.230 \\ 0.334 & 1.146 & 0.082 \\ 0.230 & 0.082 & 0.626 \end{bmatrix} \quad (15)$$

Note the diminished off-diagonal terms! The lower triangular L_t used in the search is

$$L_t = \begin{bmatrix} 1.000 & 0.000 & 0.000 \\ 0.268 & 1.000 & 0.000 \\ 0.367 & 0.131 & 1.000 \end{bmatrix},$$

and the diagonal matrix D_t used in the search is

$$D_t = \begin{bmatrix} 4.310 & & \\ & 1.135 & \\ & & 0.626 \end{bmatrix}.$$

Determining the size of the search volume for the transformed L_t and D_t yields $\chi^2 = 1.218$. The search domain is defined as $(\boldsymbol{I}-\hat{\boldsymbol{I}})^{\mathrm{T}}(Z^{\mathrm{T}} Q_{\hat{I}} Z)(\boldsymbol{I}-\hat{\boldsymbol{I}}) < \chi^2$, for $\boldsymbol{I}$ integer. The final and fixed ambiguities are

$$\bar{\boldsymbol{I}} = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}.$$

Comparing this vector with $\hat{\boldsymbol{I}}$ in (9), we notice that the final integer result could not be derived from the float values; this is due to the non-zero correlations between the individual ambiguities.

With this we end the simple numerical example demonstrating the mechanism of the LAMBDA method.

In the next issue of this periodical, we will illustrate the concept of receiver autonomous integrity monitoring (RAIM), and horizontal and vertical protection levels as applied especially in aviation.

Manufacturers

Matlab is a product of **The MathWorks, Inc.**, Natick, Massachusetts, USA.

Additional Resources

[1] de Jonge, P. J., and C. C. J. M. Tiberius, "The LAMBDA Method for Integer Ambiguity Estimation: Implementation Aspects," *Delft Geodetic Computing Centre LGR series, No. 12,* Delft, Netherlands, 1996

[2] Lenstra, A. K., and H. W. Lenstra, Jr., and L. Lovász, "Factoring polynomials with rational coefficients," *Mathematische Annalen* 261 (4): 515–534, (1982)

[3] Strang, G., and K. Borre, *Linear Algebra, Geodesy, and GPS*, Wellesley-Cambridge Press, Wellesley, Massachusetts, USA, 1997

[4] Teunissen, P. J. G., "The least-squares ambiguity decorrelation adjustment: a method for fast GPS integer ambiguity estimation," *Journal of Geodesy,* 70: 65–82, 1995

Author

Originally graduated as a chartered surveyor, **Kai Borre** obtained his Ph.D. in geodesy from Copenhagen University, and a Doctor of Technology degree from Graz University of Technology. He has been a full professor in geodesy at Aalborg University since 1976.

For more than 30 years Borre has conducted research and performed education in the area of satellite based positioning. In 1996 he established the Danish GPS Center and since 2000 has been head of a two-year international M.Sc. program in GPS technology.

Borre is a coauthor of the widely used textbooks, *Linear Algebra, Geodesy, and GPS* and *A Software-Defined GPS and Galileo Receiver; Single-Frequency Approach.*

Since the early 1990s Borre has published Matlab code for processing of GPS observations, and since 2003 he published Matlab code together with explanatory text, a very successful new pedagogical concept. IG

GPS EASY Suite II

easy13—RAIM

KAI BORRE
AALBORG UNIVERSITY

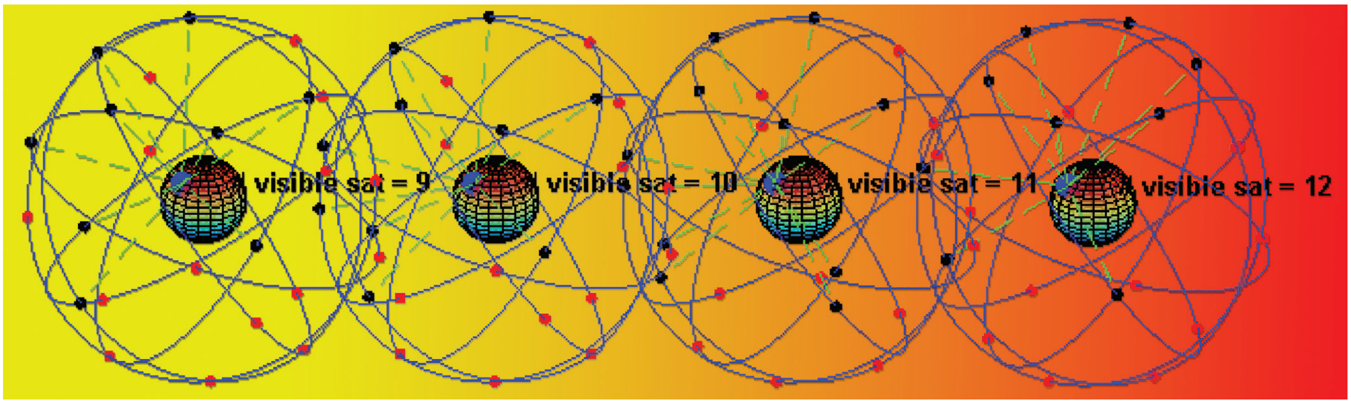


Image of GPS constellation based on public domain file from Wikimedia Commons

In this installment of the series, the author uses Matlab to illustrate key principles in receiver autonomous integrity monitoring.

The usual formulation of position determination involves four unknowns: three physical dimensions (*X,Y,Z*) and the satellite-receiver time offset. In cases where we can observe five or more pseudoranges, one might well ask if the redundant pseudoranges could be used to check the consistency among the observations —the fundamental principle behind *receiver autonomous integrity monitoring* (RAIM).

easy13 describes a technique for coping with this situation. In so doing, key concepts such as *horizontal* and *vertical protection levels* (HPL and VPL) are introduced. Necessarily, we also have to introduce some theory that motivates the procedures. (We will return to this topic with some further graphical illustrations in easy14.)

RAIM is a major technique for GNSS in many safety-critical applications. It has been with us since about 1990. Much of the material presented in the following relies on the work by B. Pervan cited in the Additional Resources section near the end of this article.

Let the 4 × 1 vector of unknowns be denoted $\boldsymbol{x}$, the $m \times 1$ vector of observations be denoted $\boldsymbol{b}$. A is an $m \times 4$ matrix and the pertinent linear observation equation is:

$$A\boldsymbol{x} = \boldsymbol{b} + \boldsymbol{e}, \quad and \quad \Sigma_b = \sigma_b^2 I. \qquad (1)$$

The vector $\boldsymbol{e}$ contains residual errors in the observations and Σ_b is the given covariance matrix for the pseudoranges.

RAIM is activated for $m \geq 5$. Presently there is no standardized RAIM method; so, we choose to present the simplest RAIM fault detection based on the residual norm $\|\boldsymbol{e}\|$.

We define the *position error* as

$$\delta\boldsymbol{x} = \boldsymbol{x} - \hat{\boldsymbol{x}} = \boldsymbol{x} - (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\boldsymbol{b} = \boldsymbol{x} - (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}(A\boldsymbol{x} - \boldsymbol{e}) = \qquad (2)$$
$$= (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\boldsymbol{e}.$$

The estimated residuals $\hat{\boldsymbol{e}}$ equal the observations $\boldsymbol{b}$ minus the estimated observations

$$\hat{\boldsymbol{e}} = \boldsymbol{b} - A\hat{\boldsymbol{x}} = (I - A(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}})\boldsymbol{b} = S\boldsymbol{b}. \qquad (3)$$

The residual vector $\hat{\boldsymbol{e}}$ is in the left nullspace of A. This means $A^{\mathrm{T}}(\boldsymbol{b} - A\hat{\boldsymbol{x}}) = 0$, which are the normal equations. The components of $\hat{\boldsymbol{e}}$ are dependent as they are computed according to (3). The corresponding covariance matrix is

$$\Sigma_{\hat{e}} = S\Sigma_b S^{\mathrm{T}} = \sigma_b^2 SS^{\mathrm{T}} = \sigma_b^2 S. \qquad (4)$$

Note that S is a projector and thus idempotent: $S = SS^{\mathrm{T}}$, and that Σ_b is diagonal, while $\Sigma_{\hat{e}}$ is a full matrix!

In order to identify the probability distribution of the residuals we need to transform the vector $\hat{\boldsymbol{e}}$ into independent components. This is done by an old trick, which implies multiplication to the left with $\frac{1}{\sigma_b}W$. The factor W comes from the Cholesky decomposition of the covariance matrix

$$\Sigma_{\hat{e}} = \left(\sigma_b W^{-1}\right)\left(\sigma_b W^{-\mathrm{T}}\right).$$

The transformed residual vector

$$\hat{\boldsymbol{e}}^* = \frac{1}{\sigma_b}W\hat{\boldsymbol{e}}$$

has independent components.

Under normal conditions (small $\|\hat{\boldsymbol{e}}\|$) the weighted sum of squares is

$$\hat{\boldsymbol{e}}^{\mathrm{T}}\frac{1}{\sigma_b^2}W^{\mathrm{T}}W\hat{\boldsymbol{e}} = (\hat{\boldsymbol{e}}^{*})^{\mathrm{T}}\hat{\boldsymbol{e}}^{*}. \quad (5)$$

The vector $\hat{\boldsymbol{e}}^{*}$ is gaussian and independent and identically distributed with zero mean and variance 1:

$$\left\|\hat{\boldsymbol{e}}^{*}\right\| : \chi^2_{m-4}, \quad m > 4. \quad (6)$$

Figure 1 graphically illustrates some basic RAIM states, which eventually (as illustrated in Figure 4) become four possible outcomes or "cases" experienced when using RAIM techniques: normal (detected) error condition (NC), missed detection (MD), detection failure (DF), and false alarm (FA).

A residual threshold can be set analytically using (6) to achieve any desired probability of false alarm under normal error conditions:

$$P(\mathrm{FA}\,|\,\mathrm{NC}) = P(\left\|\hat{\boldsymbol{e}}^{*}\right\| > R\,|\,\mathrm{NC}) = \quad (7)$$

$$= \frac{1}{2^{(m-4)/2}\,\Gamma\left(\frac{m-4}{2}\right)} \int_{R^2/\sigma_b^2}^{\infty} s^{\left(\frac{m-4}{2}-1\right)} e^{-s/2}\, ds.$$

Given the values of m - 4 and P(FA | NC) we may solve (7) for the residual threshold R. The situation is depicted in **Figure 2**. The Matlab code (M-code) for plotting this figure is contained in the file "fap.m," which can be found at the easy2 webpage < http://kom.aau.dk/~borre/easy2/easy13>.

In Figure 1 a horizontal line constraint is drawn to represent the protection level a. Note that, for small failure magnitudes, it is possible for the accuracy specification not to be breached.

In the event that the position error $\delta\boldsymbol{x}$ exceeds a predefined protection level a, but $\|\hat{\boldsymbol{e}}\| < R$ — the residual threshold R determined from equation (7) — a *missed detection* has occurred, case II, see Figures 1 and 4. The corresponding probability is defined as

$$P(\mathrm{MD}) = P(\left\|\hat{\boldsymbol{e}}\right\| < R, \left\|\delta\boldsymbol{x}\right\| > a). \quad (8)$$

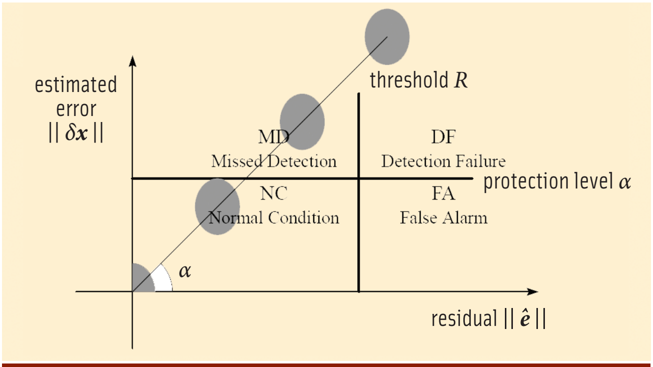
In general a condition between $\|\hat{\boldsymbol{e}}\|$ and $\|\delta\boldsymbol{x}\|$ will exist. We must quantify the degree of this correlation in order to demonstrate the integrity monitoring capability of RAIM-based fault detection. The result is given later in equation (13).

According to equation (2) the residual $\hat{\boldsymbol{e}}$ and the position error δx will scale proportionally, with the factor $(A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}$. Hence, the normal condition (NC) confidence ellipse will slide up the *failure mode axis* with slope α.

In (2) the position error

$$\delta x = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}e$$

is defined in a 3-D Cartesian coordinate system (X,Y,Z). However, for practical use a local topocentric coordinate system $\delta\boldsymbol{x}_{ENU} = (e, n, u)$ is more appropriate. A position vector at (φ, λ, h) given in the (X,Y,Z) system can be transformed into the east, north, up (e, n, u) system through multiplication by the orthogonal transformation matrix F:



estimated error $\|\delta\boldsymbol{x}\|$
threshold R
MD Missed Detection
DF Detection Failure
protection level α
NC Normal Condition
FA False Alarm
α
residual $\|\hat{\boldsymbol{e}}\|$

FIGURE 1 Basic RAIM states

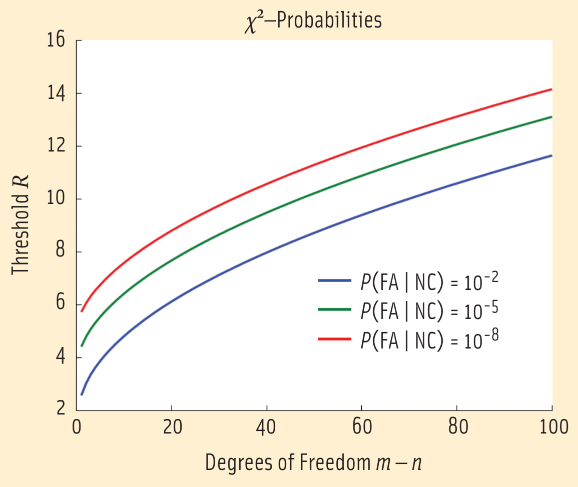


FIGURE 2 Probability of a false alarm

$$F = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{bmatrix}. \quad (9)$$

In the following discussion we only consider the three coordinates (X,Y,Z). So we delete the last column of A and get a new matrix $A_0 = A(:, 1{:}3)$. Similarly we delete the last element of $\delta\boldsymbol{x}$ and define $\delta\boldsymbol{x}_0 = \delta\boldsymbol{x}(1{:}3)$; hence:

$$\delta\boldsymbol{x}_{ENU} = \begin{bmatrix} e \\ n \\ u \end{bmatrix} = F\delta\boldsymbol{x}_0 = F(A_0^{\mathrm{T}}A_0)^{-1}A_0^{\mathrm{T}}\boldsymbol{e} = M\boldsymbol{e}. \quad (10)$$

Note that rows 1 and 2 of the $3 \times m$ matrix M relate to easting and northing. Finally we define $M_0 = M(1{:}2, 1{:}2)$.

Imagine now a *failure* of magnitude β in satellite i (β is placed as the ith component):

$$\boldsymbol{e} = [\,0 \cdots 0 \quad \beta \quad 0 \cdots 0\,]^{\mathrm{T}}. \quad (11)$$

We compute the norm squared for this special choice of $\boldsymbol{e}$:

$$\left\|\delta\boldsymbol{x}_{EN}\right\|^2 = \boldsymbol{e}^{\mathrm{T}}M_0^{\mathrm{T}}M_0\boldsymbol{e}.$$

The many zeros in $\boldsymbol{e}$ simplifies this to

$$\left\|\delta \boldsymbol{x}_{EN}\right\|^2 = (m_{1i}^2 + m_{2i}^2)\beta^2 .$$

From (3) we recall $\hat{\boldsymbol{e}} = S\boldsymbol{b}$ or

$$\left\|\hat{\boldsymbol{e}}\right\|^2 = \hat{\boldsymbol{e}}^{\mathrm{T}}\hat{\boldsymbol{e}} = \boldsymbol{b}^{\mathrm{T}}S^{\mathrm{T}}S\boldsymbol{b} = s_{ii}\beta^2$$

as $S^{\mathrm{T}}S = S$. The diagonal entry (i,i) of S is called s_{ii}. Now

$$\left\|\delta \boldsymbol{x}_{EN}\right\|^2 = \frac{m_{1i}^2 + m_{2i}^2}{s_{ii}}\left\|\hat{\boldsymbol{e}}\right\|^2$$

or

$$\left\|\delta \boldsymbol{x}_{EN}\right\| = \sqrt{\frac{m_{1i}^2 + m_{2i}^2}{s_{ii}}}\left\|\hat{\boldsymbol{e}}\right\| = \alpha_i\left\|\hat{\boldsymbol{e}}\right\|. \qquad (12)$$

This is the equation for a straight line through the origin and with slope α_i. The slope α_i of the failure mode axis related to satellite i is computed as

$$\alpha_i = \sqrt{\frac{m_{1i}^2 + m_{2i}^2}{s_{ii}}}. \qquad (13)$$

The slope values are computed for all $i = 1, \ldots, m$, and the corresponding lines are depicted in **Figure 3**. The PRNs in this figure are the ones included in easy2 (computation of a satellite's position from an ephemeris).

The likelihood that the RAIM algorithm may detect an observational error depends on the satellite geometry. A poor geometry does not necessarily indicate observational errors, but if errors are present they may be difficult to detect.

The slope α_i provides a measure of the difficulty in accurately detecting a fault in presence of noise: the larger the slope, the more difficult it is to detect the fault.

The failure mode axes in Figure 3 through the origin with slope α_i are given exclusively from the geometry determined by the satellites and the receiver. The mode axis with maximum value of α_i is called α_{max} and the HPL is defined as

$$HPL = \alpha_{\max}\,\sigma_0 \qquad (14)$$

where σ_0 is the standard deviation of the pseudoranges

$$\sigma_0 = \sqrt{\frac{\hat{\boldsymbol{e}}^{\mathrm{T}}\,\Sigma_b^{-1}\,\hat{\boldsymbol{e}}}{m-4}}.$$

At <http://www.nstb.tc.faa.gov/Terms.html> readers may find the following definition of HPL: The Horizontal Protection Level is the radius of a circle in the horizontal plane with its center being at the true position which describes the region that is assured to contain the indicated horizontal position.

The resulting RAIM fault detection algorithm is a simple one: Check the residual statistic to see if it is larger than the threshold R. If so, a system failure is declared. Given this simple algorithm, four outcomes are possible, refer to Figures 1 and 4.

Under a *normal condition* (NC), the position error $||\delta \boldsymbol{x}||$ does not exceed the protection level a, and the residual is smaller than the threshold R, as in case III. If the position error does not exceed the protection level a, but the residual is larger than

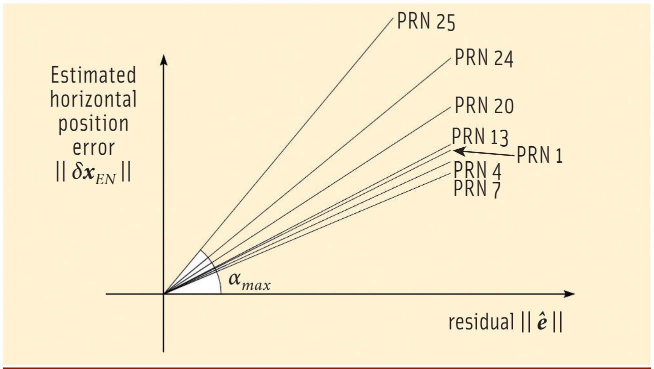


FIGURE 3 Characteristic slopes for seven visible satellites

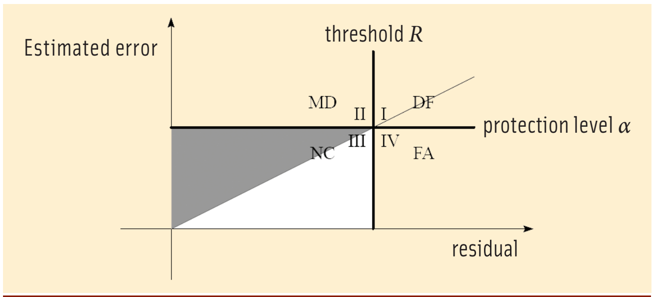


FIGURE 4 RAIM status, repeated. We aim at having plots in the gray area.

the threshold R, a *false alarm* (FA) has occurred, which is case IV. When both protection level and residual threshold have been breached, a *detection failure* (DF) has occurred — case I. Finally, a *missed detection* (MD) happens when the position error $||\delta \boldsymbol{x}||$ is larger than the protection level a, but the residual is smaller than the threshold R; that is case II.

In the general case, of course, more than one failure mode exists, that is, $\boldsymbol{e}$ in (11) has more than one non-zero component. However, this presentation does not deal with that case. In their book (see Additional Resources), R. G. Brown and P. Y. C. Hwang investigate RAIM in case of non-uniform weighted observations and multiple faults.

Because the horizontal protection level depends on satellite geometry, it must be computed for each epoch and each position. If the HPL is below the protection level, RAIM is said to be available for that epoch.

Referring to **Figure 4**, we may introduce inequalities, which characterize each region in the figure. For notational reasons, in addition to the predefined protection level a, we introduce the obvious new variables $\boldsymbol{x} = ||\hat{\boldsymbol{e}}||$ (norm of residuals) and $y = ||\delta \boldsymbol{x}_{EN}||$ (horizontal position error):

case I	upper part	$a < y < x$
	lower part	$a < x < y$
case II		$y < a < x$
case III	upper part	$y < x < a$
	lower part	$x < y < a$
case IV		$x < a < y$

A desired result will plot in the case III area, upper part, which is called *normal operation*. The foregoing description may be summarized into the following procedure for determining HPL:

Input to RAIM: The variance σ_b^2 of a pseudorange observation, the coefficient matrix A of the linearized least-squares observation equations, and the maximum allowable probabilities for a false alarm $P(\text{FA})$ and a missed detection $P(\text{MD})$.

Output of the algorithm: Horizontal protection level (HPL), which is the radius of a circle, centered at the true position that is assured to contain the indicated horizontal position with the given probability of false alarm and missed detection.

Similarly for Vertical Protection Level (VPL). Again, an official definition (at <http://www.nstb.tc.faa.gov/Terms.html>) is: The vertical protection level is half the length of a segment on the vertical axis with its center being at the true position, which describes the region that is assured to contain the indicated vertical position.

Additional Resources

The original Easy Suite can be found online at <http://kom.aau.dk/~borre/easy/>. The complete set of Easy Suite II Matlab codes can be found in compressed ("zipped") files at <http://gps.aau.dk/~borre/easy2>.

Brown, R. G., and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, third edition, John Wiley & Sons, Inc., New York, 1997

de Jong, K., "Real-time integrity monitoring, ambiguity resolution and kinematic positioning with GPS," *Proceedings of 2nd European Symposium on GNSS'98,* Toulouse, France, pp. VIII07/1-VIII07/7, 1998

Kaplan, E., and C. J. Hegarty (editors), *Understanding GPS, Principles and Applications,* second edition, Artech House, Boston and London, 2006

Nikiforov I., and B. Roturier, "Advanced RAIM algorithms: First results," ION GNSS2005, 18th International Technical Meeting of the Satellite Division of the Institute of Navigation, Long Beach, California, USA

Pervan, B. S., *Navigation Integrity for Aircraft Precision Landing Using the Global Positioning System.* Stanford University, Stanford, California, USA, 1996

Author

Originally graduated as a chartered surveyor, **Kai Borre** obtained his Ph.D. in geodesy from Copenhagen University, and a Doctor of Technology degree from Graz University of Technology. He has been a full professor in geodesy at Aalborg University since 1976.

For more than 30 years Borre has conducted research and performed education in the area of satellite based positioning. In 1996 he established the Danish GPS Center and since 2000 has been head of a two-year international M.Sc. program in GPS technology.

Borre is a coauthor of the widely used textbooks, *Linear Algebra, Geodesy, and GPS* and *A Software-Defined GPS* and *Galileo Receiver; Single-Frequency Approach*.

Since the early 1990s Borre has published Matlab code for processing of GPS observations, and since 2003 he published Matlab code together with explanatory text, a very successful new pedagogical concept. IG

GPS EASY Suite II

easy14—EGNOS-Aided Aviation

KAI BORRE
AALBORG UNIVERSITY

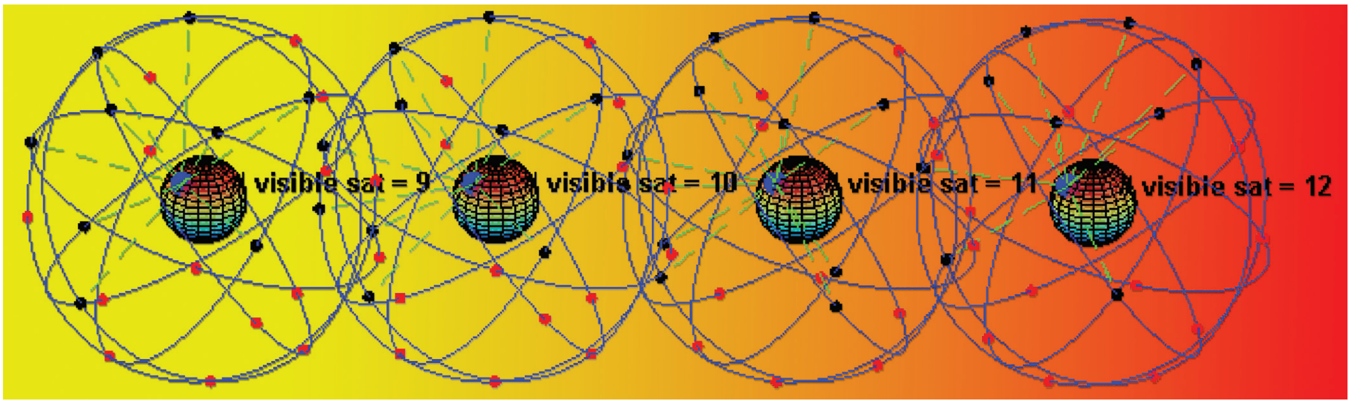


Image of GPS constellation based on public domain file from Wikimedia Commons

In this installment of the series, the author uses Matlab to illustrates aspects of aviation navigation using EGNOS signals to augment GPS.

Some GPS applications generate long time series of position estimates. Most often, an easy way of getting a quick overview on the recorded data is to plot them. However, it may be difficult from a plot to quantify statistical measures such as *mean value, standard deviation,* and *circular error probable* (CEP).

easy14 continues the aviation-oriented discussion begun in the July/August 2009 issue of *Inside GNSS*, drawing on use of the European Geostationary Navigation Overlay Service (EGNOS), a satellite-based augmentation system (SBAS). The files for the corresponding MATLAB codes can be found online; details included in the Additional Resources section near the end of this article.

Our demonstration sample uses data collected on August 20, 2008, over a 4.2-hour period with a static GPS L1 receiver with 12 code and carrier channels and two optional SBAS channels, located at a site south of Aalborg, Denmark. This receiver is a commercial one capable of handling EGNOS data. However, dedicated EGNOS receivers exist that exploit the data optimally and from which better-looking plots can be generated.

The EGNOS-corrected positions were computed by using a MATLAB implementation done at the Danish GPS Center (DGC). For a given alert limit *a*, EGNOS provides an integrity measure that tells the user to use or not to use the position in question.

The EGNOS corrections can be obtained in several ways: directly via the geostationary satellites listed in **Table 1**, or from a service called Signal in Space through the Internet (SISNeT), or through other services. In the present case we apply a post-processing mode which relies on the EGNOS Message Server (EMS) on-line at <ftp://131.176.49.48/pub/>

In polar regions users may have difficulties in receiving signals from the geostationary satellites and, hence, the Internet version becomes useful. However, the Internet is not available near the poles too often!

We shall make a small digression to investigate the magnitude of the elevation angle *h* of a (geostationary) satellite as a function of the receiver site. Let a ground receiver have coordinates (φ_E, λ_E) and let the sub-satellite coordinates of any satellite be (φ_S, λ_S). The north pole, the receiver position *E*, and the sub-satellite point of *S* make a spherical triangle on a unit sphere with sides $90^\circ - \varphi_E$, $90^\circ - \varphi_S$, and the intersecting angle $\lambda_E - \lambda_S$.

Applying the spherical cosine law on this triangle yields the following expression for the latitude φ reduced to the meridian of *E*

$$\cos\varphi = \cos\varphi_E \cos\varphi_S \cos(\lambda_E - \lambda_S) + \sin\varphi_E \sin\varphi_S .$$

Specializing to a geostationary satellite with $\varphi_S = \varphi_G = 0$ gives

$$\cos\varphi = \cos\varphi_E \cos(\lambda_E - \lambda_G). \qquad (1)$$

Our data are collected at $(\varphi_E, \lambda_E) = (57^\circ, 10^\circ)$; if we received the EGNOS corrections from the AOR-E (PRN 120) satellite, we would have

$$\cos\varphi = \cos 57^\circ \cos(10^\circ - (-15.5^\circ))$$

or a reduced latitude $\varphi = 60.6^\circ$.

Next we assume that our site E and the geostationary satellite G are in the same meridian plane. The Earth radius is $a_E = 6700$ kilometers and the distance between the geostationary satellite and the Earth center is $a_G = 42486$ kilometers. With reference to **Figure 1** we get, in a spherical approximation, for triangle *OPG*

$$\frac{a_G}{\sin(\pi/2 + h)} = \frac{a_E}{\sin\left(\pi/2 - (h + \varphi)\right)}. \qquad (2)$$

Some rewriting involving the sum relation yields

$$h = \arctan\left(\frac{\left|\cos\varphi - \frac{a_E}{a_G}\right|}{\sin\varphi}\right) \qquad (3)$$

The elevation angle h equals zero for $\varphi = 80.9^\circ$; for $\varphi = 60.6^\circ$ we get $h = 20.9^\circ$, (refer to **Figure 2**).

The EGNOS information allows us to correct the observed pseudoranges for atmospheric delays and other error sources. A fast EGNOS correction is computed from navigation messages 2–5 that contain the so-called fast data set. The message starts with the 2-bit issue of data PRN (IODP), an 8-bit preamble, a 6-bit message type identifier (values can be 2, 3, 4, or 5), and a 2 bit issue of data fast (IODF).

Next follows the fast data set for 13 satellites: 12 bits for the fast pseudorange correction PRC and 4 bits for the user *differential range error indicator* (UDREI). The message ends with 24 parity bits, in total 250 bits. The fast data set for the next 13 satellites is contained in the following message and so forth.

The *PRC* value is given in the interval [-256.000, +255.875] with a resolution of ⅛ meter.

The range-rate correction *RRC* of the fast correction is computed as the difference between the current *PRC* and the previous one divided by the time interval between the two values. The time of applicability t_{of} is identical to the transmission time of the first bit in the message.

The fast correction to the pseudorange for a given satellite identified by its pseudorandom noise code number (PRN) is

$$PR_{\text{corrected}}(t) = PR_{\text{observed}}(t) + PRC(t_{of}) + RRC(t_{of})(t - t_{of}). \qquad (4)$$

The *RRC* computation must time-out if there is no *PRC* for eight seconds.

The complete computation of the EGNOS correction involves calculation of an ionospheric correction and a possible tropospheric correction. The variance of the ith observation is composed of four variances: a fast and long term contribution σ^2_{flt}, a tropospheric correction σ^2_{tropo}, an ionospheric correction σ^2_{iono}, and finally receiver noise and multipath σ^2_{air}:

$$\sigma^2_i = \sigma^2_{\text{flt}} + \sigma^2_{\text{tropo}} + \sigma^2_{\text{iono}} + \sigma^2_{\text{air}}. \qquad (5)$$

Typical values for a satellite with high elevation are $\sigma_{\text{flt}} = 0.26$ m, $\sigma_{\text{tropo}} = 0.01$ meter, $\sigma_{\text{iono}} = 0.21$ meter, and $\sigma_{\text{air}} = 3.26$ meter, and for a low elevation satellite $\sigma_{\text{flt}} = 0.55$ meter, $\sigma_{\text{tropo}} = 0.10$ meter, $\sigma_{\text{iono}} = 0.91$ meter, and $\sigma_{\text{air}} = 3.28$ meter.

Figure 3 shows the time series (deviations from mean values) for the three coordinates East, North, and Up (E, N, U) at the location E. Positions computed from the raw pseudoranges (black line) plot as noisy curves.

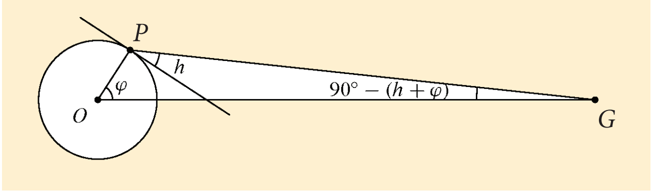


FIGURE 1 Elevation angle h to geostationary satellites as seen from the Earth's surface as a function of latitude φ, spherical approximation

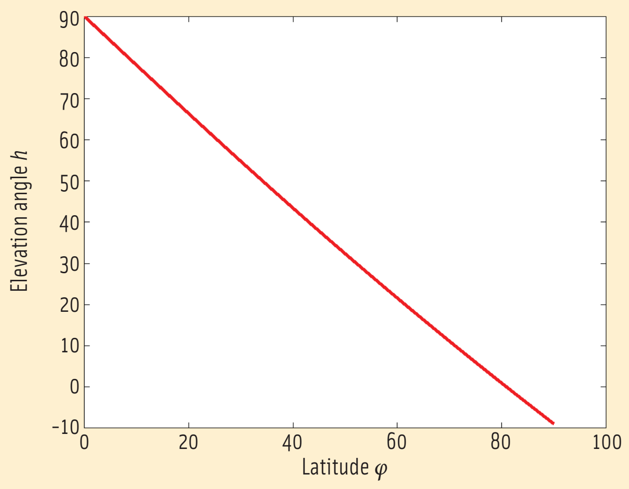


FIGURE 2 Plot of elevation angle h to geostationary satellites as seen from the Earth's surface as a function of latitude φ

Satellite Name	Longitude l_G	PRN
AOR-E	15.5° W	120
Artemis	21.5° E	124
IOR-W	25.0° E	126

TABLE 1. EGNOS geostationary satellites

Next we applied a MATLAB EGNOS correcting code, developed at DGC, to the raw observations; the result is the less noisy curves (red). No smoothing is applied. A further quantification of the graphs in **Figure 3** are added as **Table 2**. **Figure 4** contains a plot of the corresponding horizontal positions.

We have created a script easy141 that gives a recipe for computing *horizontal protection levels* (HPLs) and *vertical protection levels* (VPLs) for aircraft approaches to airports. We start from the coefficient matrix A for a single epoch

$$A = \begin{bmatrix} -\frac{X^1 - X}{\rho^1} & -\frac{Y^1 - Y}{\rho^1} & -\frac{Z^1 - Z}{\rho^1} & 1 \\ -\frac{X^2 - X}{\rho^2} & -\frac{Y^2 - Y}{\rho^2} & -\frac{Z^2 - Z}{\rho^2} & 1 \\ & \vdots & & \\ -\frac{X^s - X}{\rho^s} & -\frac{Y^s - Y}{\rho^s} & -\frac{Z^s - Z}{\rho^s} & 1 \end{bmatrix} \qquad (6)$$

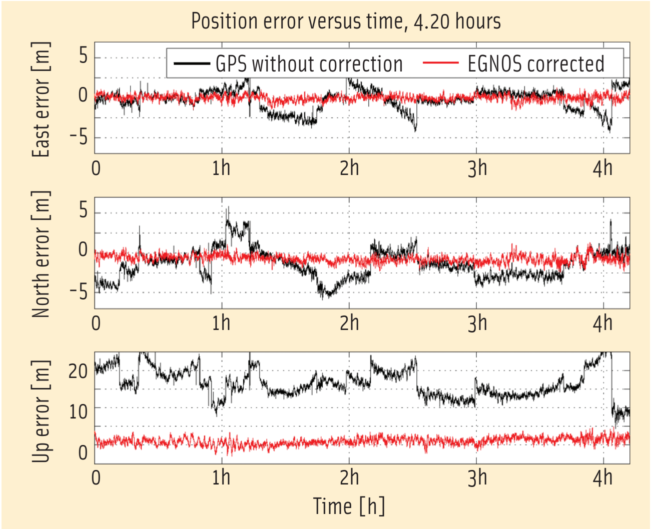


FIGURE 3 4.2 hours of data collected by a static GPS/SBAS receiver. The figure shows east, north, and up as computed from raw pseudoranges (black) and from DGC EGNOS implemented correcting code (red)

	CEP 50% [m]	CEP 95% [m]	s_c [m]	s_n [m]	$mean_e$ [m]	$mean_n$ [m]
Raw Pseudoranges	1.8	3.8	1.3	1.8	-0.2	-1.4
DGC EGNOS-corrected	0.5	1.1	0.4	0.5	0.0	-0.7

TABLE 2. Circular errors probably (CEP), standard errors, and mean values of position for raw pseudoranges collected by a commercial receiver. Also shown are EGNOS corrections as computed by DGC MATLAB code. The numbers are derived from the 4.2-hour data series shown in Figure 3. The cut-off angle is 5° elevation.

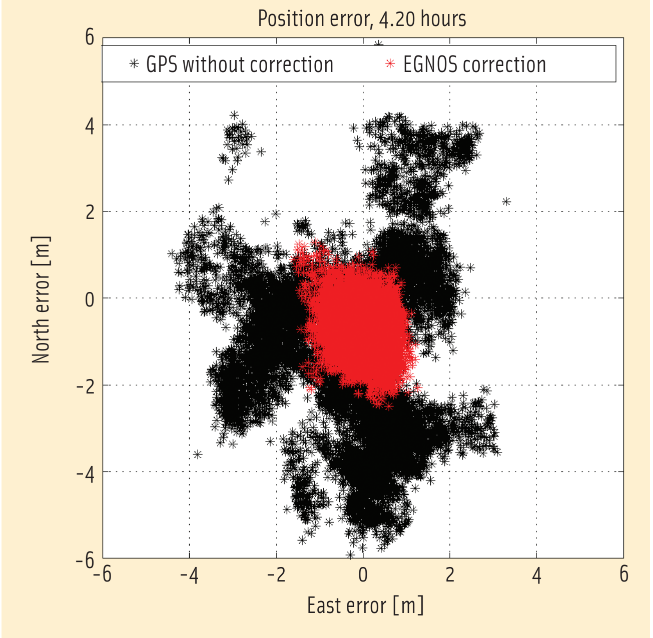


FIGURE 4 4.2 hours of data collected by a Thales DG16 receiver. The figure shows horizontal positions in meters as computed from raw pseudoranges (black) and from DGC EGNOS implemented correcting code (red)

The necessary details may be found in the script.

The larger script easy14 starts from a transformed version of *A*. The main change is to observe that $\frac{X^1 - X}{\rho^1} = \cos el^i \cos az^i$ where el^i means the elevation angle and az^i the azimuth of PRN *i* as seen from the receiver position.

The first three columns contain the so-called Euler angles between the line-of-sight and the *(X,Y,Z)* axes. However, we need the corresponding angles between the line-of-sight and the topocentric *(E,N,U)* system. The three columns are transformed from *(X,Y,Z)* to *(E,N,U)* by post-multiplication with F^T. For further discussion of this point, see equation (9) in easy13.

The product $B = A(:, 1:3)\,F^T$, augmented with a column of ones, can also be computed directly as

$$B = \begin{bmatrix} \cos el^1 \cos az^1 & \cos el^1 \sin az^1 & \sin el^1 & 1 \\ \cos el^2 \cos az^2 & \cos el^2 \sin az^2 & \sin el^2 & 1 \\ & \vdots & & \\ \cos el^s \cos az^s & \cos el^s \sin az^s & \sin el^s & 1 \end{bmatrix}. \qquad (7)$$

The next steps are described by simple expressions but involve heavy computational burdens. A weight matrix *W* for the observed pseudoranges is defined in terms of the variances σ_i^2:

$$W = \operatorname{diag}[\sigma_1^{-2} \quad \sigma_2^{-2} \quad \dots \quad \sigma_s^{-2}]. \qquad (8)$$

$$\Sigma = (B^T W B)^{-1}.$$

The 2 × 2 block matrix of the covariance matrix Σ containing the entries $\Sigma_{11} = \sigma_1^2$, $\Sigma_{22} = \sigma_2^2$, and $\Sigma_{12} = \Sigma_{21} = \sigma_{12}$ constitute the coefficients in a quadratic form

$$\sigma_1^2 x^2 + 2\sigma_{12} xy + \sigma_2^2 y^2 = 1.$$

This quadratic form defines a confidence ellipse. The largest eigenvalue λ_1 of this quadratic form is the semi-major axis of the confidence ellipse for the position solution

$$\lambda_1 = \frac{1}{2}\left(\sigma_1^2 + \sigma_2^2 + \sqrt{(\sigma_1^2 + \sigma_2^2)^2 - 4\left(\sigma_1^2\ \sigma_2^2 - \sigma_{12}^2\right)}\right),$$

Equation (9.78) in the book by G. Strang and K. Borre, listed in Additional Resources, provides further context for the preceding step.

The expression for λ_1 can be rewritten as

$$\lambda_1 = \frac{\sigma_1^2 + \sigma_2^2}{2} + \sqrt{\left(\frac{\sigma_1^2 - \sigma_2^2}{2}\right)^2 + \sigma_{12}^2}. \qquad (9)$$

This version is found in the minimum operational performance standards (MOPS) for GPS/Wide Area Augmentation System (WAAS) airborne equipment published in *RTCA/DO-229C*, page J.1.

The EGNOS standard defines two magnification factors $K_H = 6$ and $K_V = 5.33$. Remember that $\Sigma_{33} = \sigma_{33}$ and finally we obtain:

$$\text{HPL} = K_H \sqrt{\sigma_1}$$
$$\text{VPL.} = K_V\, \sigma_{33}$$

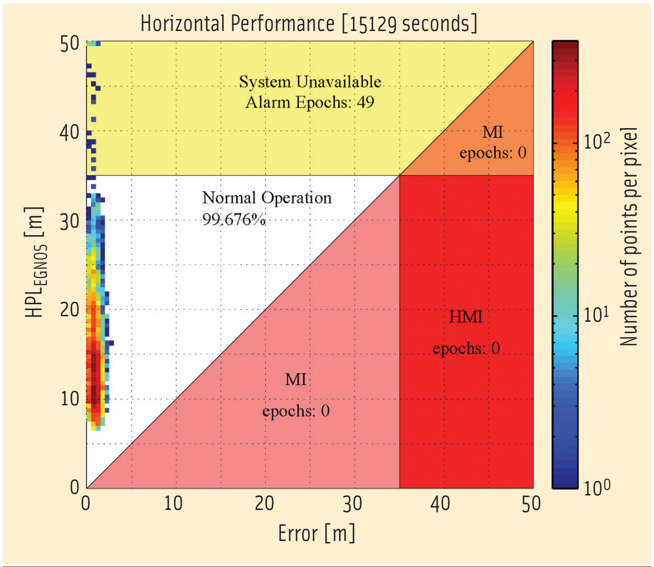


FIGURE 5 Positions depicted in Figure 3 are plotted versus horizontal protection levels. Integrity fails in the region marked HMI, hazardously misleading information. In the upper area marked MI, integrity fails and the system is unavailable. HPL failures happen in the lower area marked MI. In short: *USE* positions depicted in the white area, *DO NOT USE* positions depicted in colored areas

Typical values are: HPL = 15 and VPL = 6. The EGNOS document also introduces its own non-precise HPL value, which is 1.03 times larger than the HPL used here. Everything is implemented in easy141.

Contemporary positioning theory uses the following four concepts that enter into almost every aviation specification *accuracy, integrity, continuity,* and *availability.*

The first concept, accuracy, fits our intuition well as it measures the difference between the (corrected) computed position and the true position. An SBAS implementation is obliged to quantify the accuracy of a wide-area differentially corrected position solution. Accuracy may be estimated by the difference between a receiver position and the true position which, for CEP 95%, is only exceeded 5 percent of the time in the absence of system failures.

Integrity risk is defined as the probability that the SBAS exceeds either the horizontal or vertical alert limits (HALs or VALs) *and* the system alert is silent beyond the time-to-alarm. EGNOS has been designed for a six-second time-to-alarm.

Continuity and availability are expressed in global terms. If a user experiences a system outage due to signal blockage, from a global perspective this is not a loss in continuity of the system — it is a local phenomenon.

Clearly, we need to distinguish between issues related to availability and continuity and what is availability of the system from a user's perspective.

A sophisticated plot was developed at Stanford University in 1998. The histogram of **Figure 5** reports the horizontal system metrics provided by an EGNOS implementation done at DGC for at static user at Aalborg.

The EGNOS concept only involves an alert limit a. Knowing the value of a we can draw a half line from origin through (a,a) which partly limits the white area in which the attractive bins must lie!

The horizontal axis indicates the error $\sqrt{e^2 + n^2}$ in the SBAS navigation solution with respect to the surveyed antenna location. The vertical axis indicates the protection level computed for each and every position solution.

Each bin tabulates the number of occurrences of a specific pair (x,y) = (error, protection level), and the color indicates the total number of epochs in which that pair occurs. Note that the color scale is logarithmic and the bins are quantized into 0.25-meter squares.

In case the HPL is larger than the alarm limit a, we experience an integrity failure. The true error should always be less than the HPL. The long-term availability requirement of the SBAS is 99.9 percent and, hence, at least 999 out of 1,000 points should lie within the "Normal Operation" (USE) region.

In the Figure 5 results, the system maintained 99.7 percent availability in horizontal positioning and the system was unavailable in 49 epochs out of 15,129 that is 0.3 percent; note that (99.7 + 0.3) % = 100%.

Availability can only be associated with geostationary satellites. For users employing the Internet to obtain the EGNOS information, everything changes. In the present study we use EMS and not the real-time service SISNeT.

Figure 6 (on page 72) shows the vertical system performance corresponding to the horizontal data presented previously. The SBAS correction in the vertical dimension has poorer performance than the horizontal due to its weaker geometry; the alert limits are a = 12, 20 meters. The 12-meter level corresponds to Category I landing and the 20-meter level to instrument precision with vertical guidance (IPV). In this scenario, the positioning system met all three safety metrics (accuracy, integrity, and continuity) with an availability of 99.061 percent.

Acknowledgment

Kostas Dragūnas added several changes to the original *M*-files vplstat and hplstat written by Todd Walter of Stanford University. The files are included in the Easy Suite II files (see Additional Resources) with the permission of Stanford University. Users also need the bound2 file in the download directory.

Manufacturers

The data in easy14 was generated using a Thales DG 16 GPS receiver, from **Magellan Navigation S.A.S.**, Santa Clara, California, USA.

Additional Resources

The original Easy Suite can be found online at <http://kom.aau.dk/~borre/easy/>. The complete set of Easy Suite II Matlab codes can be found in compressed ("zipped") files at <http://gps.aau.dk/~borre/easy2>.

RTCA, Inc. (formerly Radio Technical Commission for Aeronautics), *RTCA/DO-229C, Minimum Operational Performance Standards for Global Positioning System/Wide Area Augmentation System Airborne Equipment,* 2001

GPS EASY Suite II *continued on page 72*

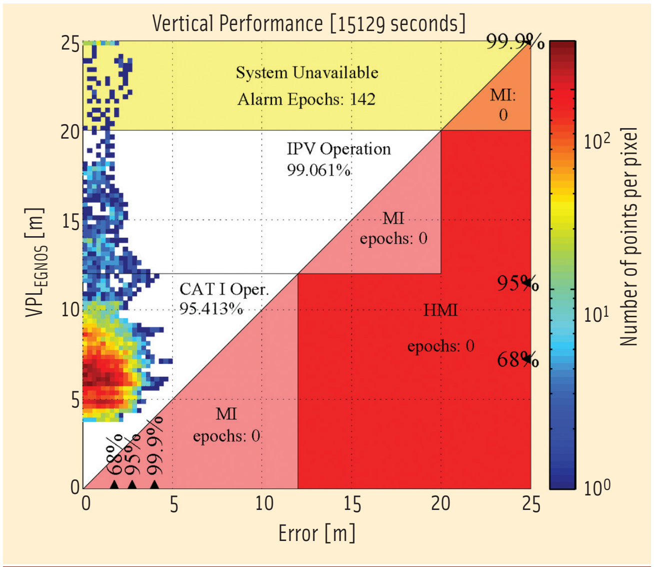


FIGURE 6 Positions depicted in Figure 10 are plotted versus vertical protection levels. Integrity failures happen in the areas marked HMI. The 20 m level corresponds to Instrument Precision with Vertical guidance (IPV). Cat I Oper. corresponds to horizontal and vertical accuracies for precision landing for Category I (decision height 200 feet). The lower and middle regions marked MI designate VPL failures. In the upper region marked MI integrity and unavailability overlap

GPS EASY Suite II *continued from page 65*
Strang, G., and K. Borre, *Linear Algebra, Geodesy, and GPS,* Wellesley-Cambridge Press, Wellesley, Massachusetts USA, 1997

Corrections to easy13

In the EASY Suite II installment in the July/August 2009 issue of *Inside GNSS,* in Figures 1 and 4, the expression "protection level α" should have been "protection level *a*."

In easy13.m the Matlab code is missing the F matrix (the transformation matrix from ECEF to ENU) in the expression for M. Hence the alpha values and HPL are wrong. Consequently Figure 3 is valid for δx_{XY} and not δx_{EN} as stated on the vertical axis.

On page 49 of the article in the paragraph following equation (10), M_o = M(1:2,1:2) should be M_o = M(1:2,:).

Author

Originally graduated as a chartered surveyor, **Kai Borre** obtained his Ph.D. in geodesy from Copenhagen University, and a Doctor of Technology degree from Graz University of Technology. He has been a full professor in geodesy at Aalborg University since 1976.

For more than 30 years Borre has conducted research and performed education in the area of satellite based positioning. In 1996 he established the Danish GPS Center and since 2000 has been head of a two-year international M.Sc. program in GPS technology.

Borre is a coauthor of the widely used textbooks, *Linear Algebra, Geodesy, and GPS* and *A Software-Defined GPS and Galileo Receiver; Single-Frequency Approach*.

Since the early 1990s Borre has published Matlab code for processing of GPS observations, and since 2003 he published Matlab code together with explanatory text, a very successful new pedagogical concept. IG

GPS EASY Suite II

easy15—Positioning Accuracy with Pseudorange & Carrier Phase

KAI BORRE
AALBORG UNIVERSITY

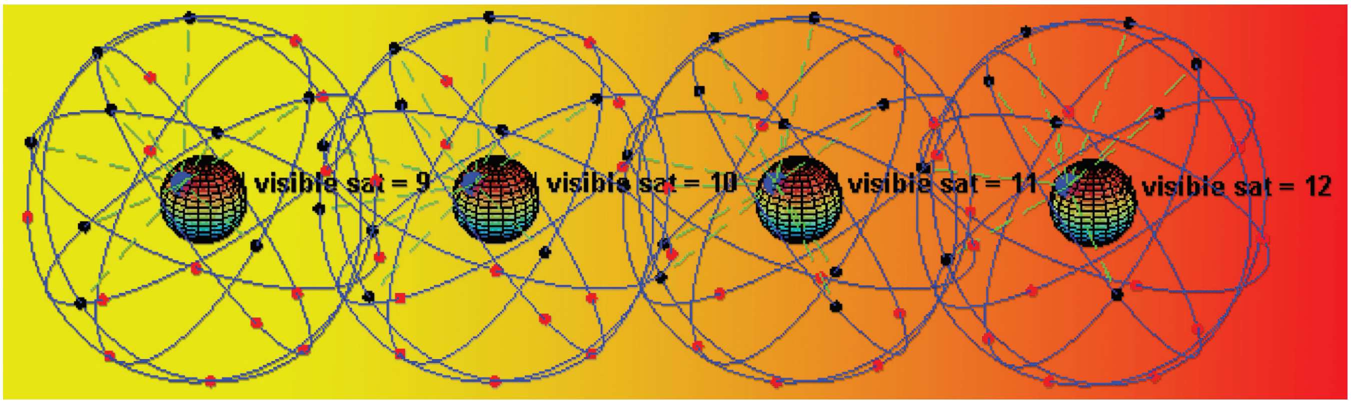


Image of GPS constellation based on public domain file from Wikimedia Commons

In this continuing series, the author describes and demonstrates the differences between position accuracy achieved using satellite-receiver pseudoranges and differential pseudoranges compared to the accuracy of carrier phase–aided positioning.

A frequently asked question about GPS is: How accurate is a GPS-based position? The experienced GPS user knows that a big difference lies between using pseudoranges alone or combining pseudoranges and carrier phases.

The following analysis is partly based on an idea raised in the book, *Global Positioning System: Signals, Measurements, and Performance,* by P. Misra and Per Enge (See Additional Resources section at the end of this article).

In order to get a quantitative answer to the question Kostas Dragūnas, a research assistant at the Danish GPS Center and now an Aalborg University Ph.D. student, recorded data simultaneously with a master and a rover receiver. In order to vary the receiver equipment, we used two dual-frequency L1/L2 GPS receivers that were different from the models employed in previous EASY Suite field campaigns. These receivers store data in a binary format called the GPS Receiver Interface Language (GRIL).

The following message was sent to the receivers before starting the observation session:

dm
em,,/msg/jps/GT,SI,R1,P1,R2,P2

Time of start was 481,860 seconds and the final epoch was at 484,395 seconds; so, 2,535 synchronous epochs of observations were recorded with a one-second epoch interval. We deleted 95 epochs because of missing data at either the rover or the master, or repeated estimation of ambiguities. This makes a continuous record of about 40 minutes.

The original observations were modified in several ways. After each ephemeris, the master receiver issued two subsequent P2 (full P/L2 carrier phases) messages (binary identical). Subsequently, the rover receiver sometimes issued two GPS Time (GT) messages — *wn* and *tow* (binary identical).

easy15 assumes data arriving in real-time from the two receivers. In reality. we read from two files, *19jan07m.log* and *19jan07r.log*, containing a mixture of observations and ephemerides. The baseline is about 1,509.3 meters long.

The actual reading of the log-files is done by the *readGrilM* and *readGrilR* functions. We are using two similar codes for reading in each log-file in order to avoid too much "bookkeeping" about where to read binary data in the files. A typical reading contains information on time of week, tracked satellites (identified by pseudorandom noise codes or PRNs), and observed pseudorange and carrier phase on the two frequencies.

The reading is complicated by the fact that new ephemerides may appear at any time in-between the observations. The ephemerides are stored in a global matrix, *EPH*. Any time a new version of an ephemeris arrives, it overwrites the old one.

As mentioned, we read from two stored files; therefore, we first have to determine the first epoch common to both receivers. Next, we determine the number of tracked PRNs at both master and rover sites for which we know the ephemerides.

When the number of common PRNs is four or more, we compute the master position using the *recposRTK* function, and all elevation angles as seen from here. We delete low-elevation PRNs and select a reference PRN. Next all observations are rearranged so that master and rover data match each other — the sequence of PRNs in the master and rover receivers are likely to be different. Ambiguities are estimated using Clyde Goad's method as described in the text by G. Strang and K. Borre, page 490, cited in Additional Resources.

New PRNs most often rise at different epochs at the master and rover. Hence, we need to omit the observations from one receiver until the PRN appears with complete data at both receivers.

We repeat the computation of the master position with good PRNs and find the ellipsoidal height h_i needed for the later computation of tropospheric delay.

We choose a Kalman filter where the state vector $\boldsymbol{x}$ has three components, namely, the baseline components (x,y,z). We initialize the covariance matrices for the vector of observations $\Sigma_{e,k}$, for the system equations $\Sigma_{\varepsilon,k}$, and for the state vector $P_{0|0}$.

Note: users should realize that a Kalman filter and its innovation vector may run smoothly and the baseline components may still be wrong. A correct ambiguity estimation leads to excellent estimates of the baseline components. Incorrect ambiguity estimates definitely lead to incorrect estimates of the baseline components while the filter works intensively on obtaining small innovations that try to push the baseline components to the correct values.

Next we read one epoch of data in the master file and in the rover file, prepare the double differenced observations, correct for tropospheric delays, and then set up the innovation vector $\boldsymbol{b} - A\boldsymbol{x}$. We update the filter, plot the result in an open figure window, and proceed to the next epoch.

This code is the closest we get to a real-time kinematic (RTK) code without having two receivers connected directly to the laptop computer's com port. If the receivers/laptop configuration can be achieved, then a user may set up the ports using the I/O facility. For one port the following code reads a data stream from the receiver into the file *legacy.tex*:

```
s = serial('COM2');
s.OutputBufferSize = 512;
s.InputBuffersize = 50000;
fopen(s);
s.BaudRate = 9600;
set(s,'TimeOut',1);
s.RecordMode = 'index';
s.RecordDetail = 'verbose';
s.RecordName = 'legacy.tex';
```

Figure 1 illustrates two levels of positioning accuracy obtained from GPS in real-time with position errors ranging from several meters to centimeters. The norm of the position error is plotted for observations taken at one-second intervals over a period of about 42 minutes.

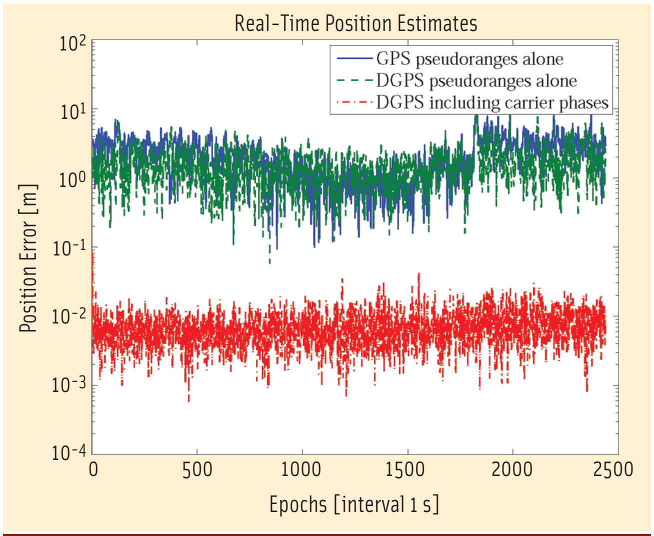


FIGURE 1 A stand-alone position and a differential position based on pseudoranges alone have position errors of a few meters, while adding carrier phase observations brings the position error down to the centimeter-level or below.

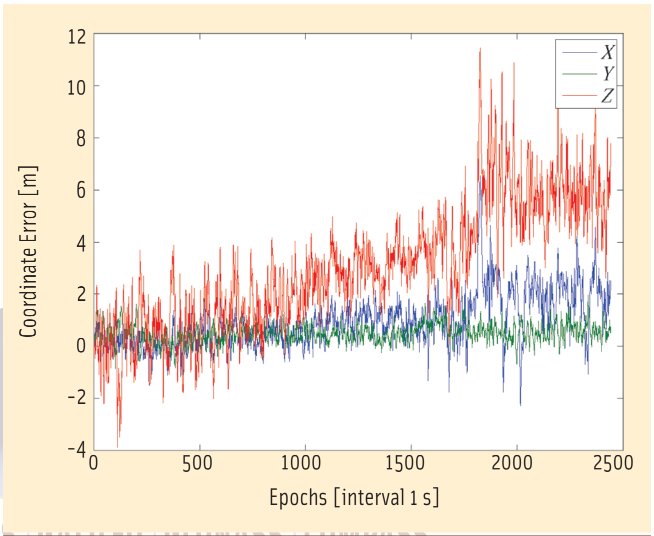


FIGURE 2 Variation of master station coordinates during 42 minutes

The raw L_1 pseudorange position accuracy is typically a few meters. Access to concurrent pseudorange observations from a GPS master receiver (at a known location) does not reduce the errors significantly; possibly it eliminates some common trends in the positions.

Finally, the full potential of the observational data is exploited by including both pseudoranges and carrier phase observations to obtain position estimates at the centimeter-level or better.

Figure 2 shows the variation over 42 minutes of master station coordinates as computed from pseudoranges alone. Typically, the *X* and *Y* coordinates show a lesser variation than the *Z* coordinate (for Aalborg, Denmark).

EASY Suite II *continued on page 65*

EASY Suite II *continued from page 51*

Manufacturer

Positioning data was gathered using two Legacy E GNSS receivers from **Topcon Positioning Systems,** Pleasanton, California, USA.

Additional Resources

[1] Misra, P. and P. Enge, *Global Positioning System: Signals, Measurements, and Performance,* 2nd edition, Ganga-Jamuna Press, Lincoln, Massachusetts, USA, 2006

[2] Strang, G. and K. Borre, *Linear Algebra, Geodesy, and GPS,* Wellesley-Cambridge Press, Wellesley, Massachusetts, USA, 1997

Author

Originally graduated as a chartered surveyor, **Kai Borre** obtained his Ph.D. in geodesy from Copenhagen University, and a Doctor of Technology degree from Graz University of Technology. He has been a full professor in geodesy at Aalborg University since 1976. For more than 30 years Borre has conducted research and performed education in the area of satellite based positioning. In 1996 he established the Danish GPS Center and since 2000 has been head of a two-year international M.Sc. program in GPS technology.Borre is a coauthor of the widely used textbooks, *Linear Algebra, Geodesy, and GPS* and *A Software-Defined GPS and Galileo Receiver; Single-Frequency Approach*. Since the early 1990s Borre has published Matlab code for processing of GPS observations, and since 2003 he published Matlab code together with explanatory text, a very successful new pedagogical concept. IG

GPS EASY Suite II

easy17—Visualizing Satellite Orbits
easy18—Computing Range and Range Rate Corrections

KAI BORRE
AALBORG UNIVERSITY

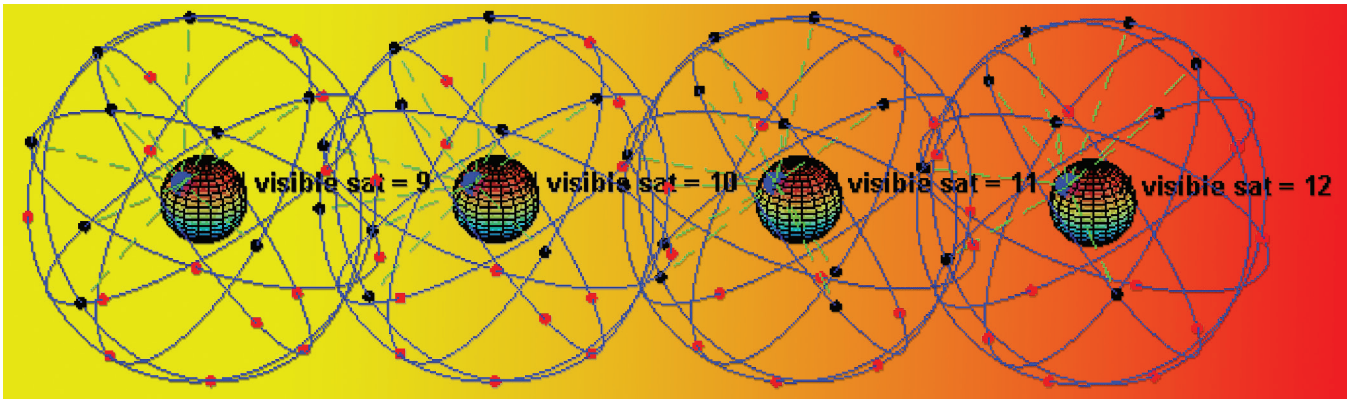


Image of GPS constellation based on public domain file from Wikimedia Commons

In the final two installments in our series, the author describes what GPS orbits would look like from various perspectives and explains how to solve for range and range-rate corrections at a GPS base station.

Newcomers often have difficulties imagining what the satellite orbits actually look like.

We say that the constellation consists of some 30 satellites orbiting in six different planes, all making an angle with the Equator of 55 degrees and rotated 60 degrees compared to the previous plane. **Figure 1** shows the situation as seen from far away in space, in what we call an *inertial frame*.

However, things get less clear if the viewer is on the surface of the rotating Earth. How do the trajectories then look? Very weird is how.

The situation is depicted in **Figure 2**.

Here we use the so-called *Earth Centered Earth Fixed (ECEF) coordinate system*. The ECEF system is in a fixed relationship with the rotating Earth. That is, a given physical point on the surface maintains its coordinates over time, except for possible movements of the crust.

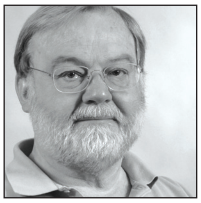
Finally, **Figure 3** shows a curve made up of the *sub-satellite points* of an arbitrary part of an orbit.

The curve is the intersection between the surface of the Earth and the line segment between the satellite and the origin. This sub-satellite curve runs within a symmetric belt on both sides of equator and is limited by northern and southern latitudes equal to the inclination angle of the orbit with the equator.

easy 18

Once I was teaching GPS to control engineers working with air traffic. A need came up for computing range and range rate

Author



Originally graduated as a chartered surveyor, **Kai Borre** obtained his Ph.D. in geodesy from Copenhagen University, and a Doctor of Technology degree from Graz University of Technology. He has been a full professor in geodesy at Aalborg University since 1976. For more than 30 years Borre has conducted research and performed education in the area of satellite based positioning. In 1996 he established the Danish GPS Center and since 2000 has been head of a two-year international M.Sc. program in GPS technology. Borre is a coauthor of the widely used textbooks, *Linear Algebra, Geodesy, and GPS* and *A Software-Defined GPS and Galileo Receiver; Single-Frequency Approach*. Since the early 1990s Borre has published Matlab code for processing of GPS observations, and since 2003 he published Matlab code together with explanatory text, a very successful new pedagogical concept.

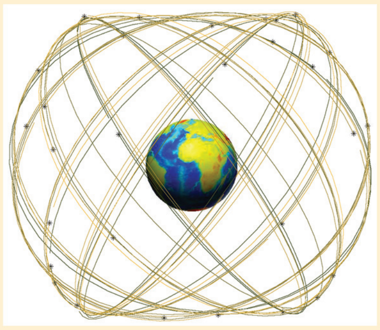


FIGURE 1 Satellite orbits as seen in inertial frame

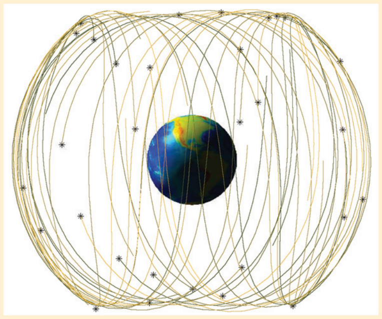


FIGURE 2 Satellite orbits as seen in Earth Centered Earth Fixed frame

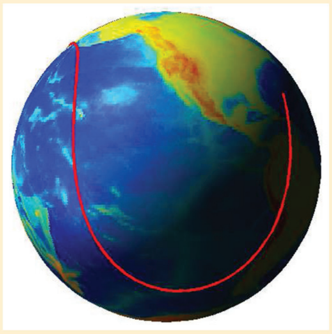


FIGURE 3 Sub-satellite points for a selected satellite

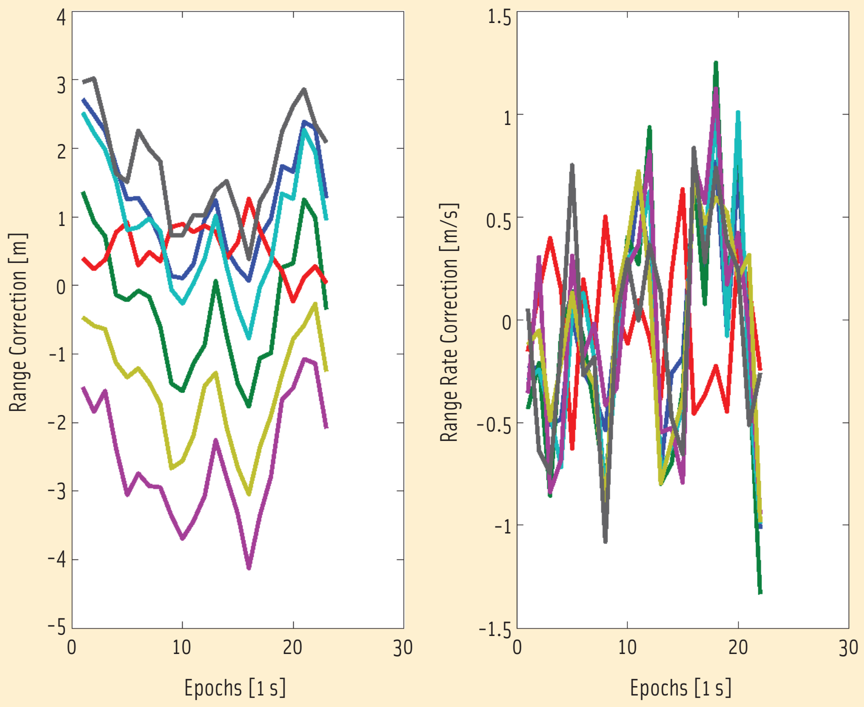


FIGURE 4 Range and range rate corrections as generated at a base station

corrections at a base station. Here is the solution.

Let ρ denote the geometric distance between the satellite and the receiver antennas, dt_i denote the receiver clock offset, dt^k the satellite clock offset, and T the tropospheric delay. Then the corrected range is computed as

$$\rho^* = \rho + cdt_i - cdt^k + T$$

and the range correction as

$$d = \rho^* - P_{obs}.$$

Figure 4 shows range and range rate corrections over 23 epochs using September 4, 2001, data.

The receiver clock offset pos(4,:) varies between 1.13×10^5 and 1.15×10^5 through the 23-second period of observations.

Acknowledgements

The help received for the EASY Suite II series from Kostas Dragūnas at the Danish GPS Center is greatly acknowledged. His help includes the sky plot, the Stanford plots, and data collection during winter time.

My thanks also go to Professor Per Enge for several invitations to stay at Stanford University's Center for Position, Navigation, and Time. I appreciated the wonderful facilities there, and all of the people I encountered. They showed so much kindness and helpfulness, and made my stays comfortable.

Finally I am very grateful to Steve Hilla for his comprehensive help in converting the manuscript from latex into Word. IG

GPS EASY Suite II

easy16—Comparing GPS Precise and Broadcast Ephemerides

KAI BORRE
AALBORG UNIVERSITY

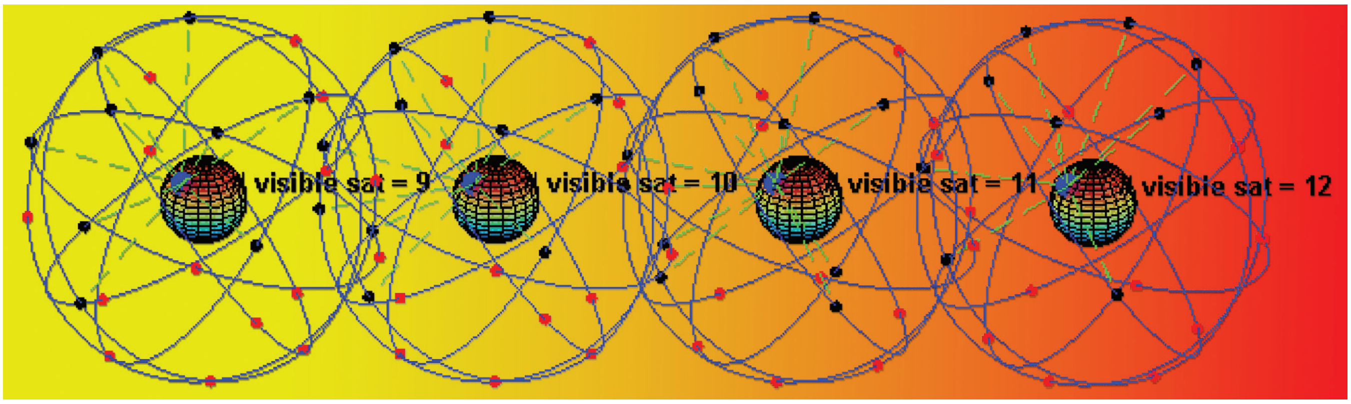


Image of GPS constellation based on public domain file from Wikimedia Commons

In this installment of the series, the author demonstrates Matlab code for estimation of ionospheric delay, multipath, and ephemeris errors.

With a given GPS receiver you get a certain positioning accuracy. So, the first question you may ask is: can I do anything to improve it?

The answer most often is: yes, go and buy a better receiver! Most ranging errors are determined by physics and you can do little to improve the situation.

However, it is interesting to analyze the information you may get from a single "one-way" range, that is, a single set of observables (P_1, Φ_1, P_2, Φ_2) related to an individual satellite seen by a single dual-frequency receiver. By so doing you also obtain an insight into the nature of the error sources — the ones that do matter and the ones that can be modeled or eliminated.

From the ephemeris of the tracked satellite we can compute the elevation angle, the ionospheric delay is estimable because the data originate from a dual-frequency receiver, the tropospheric delay is computed from a standard model, and the remaining errors we call *multipath* errors.

Finally, we estimate the orbital errors by comparing satellite positions computed from broadcast and precise ephemerides. The data sample illustrates these different error contributions very well, their variation over time, and, therefore, their dependency on elevation angle.

We select a specific one-way observation set that was acquired at a station south of Aalborg on January 19, 2007. (This is from the data set already used in easy15, published in the January/February 2010 issue of *Inside GNSS*.) We investigated PRN14 in the full session length, that is, about 42 minutes — including the 95 epochs that we omitted in the easy15 discussion because of missing data at either the rover or the master, or repeated estimation of ambiguities.

The subplots in **Figure 1** depict the satellite *elevation angle*, which varies from 18 to 33 degrees, and one-way errors in *ionosphere*, *troposphere*, and *multipath*.

When dealing with the ionospheric delay I, we assume that we have both pseudorange and carrier phase observations on L_1 and L_2 at our disposal. We estimate the delay I according to the following matrix equation:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & \lambda_1 & 0 \\ 1 & \alpha & 0 & 0 \\ 1 & -\alpha & 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \rho^* \\ I \\ N_1 \\ N_2 \end{bmatrix} = \begin{bmatrix} P_1 \\ \Phi_1 \\ P_2 \\ \Phi_2 \end{bmatrix} + \boldsymbol{e}. \qquad (1)$$

As usual we define the constant $\alpha = (f_1/f_2)^2 = I_2/I_1$. First we estimate the ambiguities on L_1 and L_2 — N_1 and N_2, respectively — as reals; incorrect values for N_1 and N_2 just change the level for I. Next we estimate I alone as

$$I = (-\lambda_1 N_1 + \lambda_2 N_2 + \Phi_1 - \Phi_2)/(\alpha - 1). \qquad (2)$$

For the tropospheric delay we use the Goad-Goodman model. The delay T ranges from 7.4 m to 4.2 meters. If the satellite passed zenith, the corresponding delay would have been about 2.5 meters.

Multipath describes the situation where signals coming from the satellite propagate along several paths to the receiver antenna. The main part of the signal radiates directly from the satellite, but part of the signal is reflected from surfaces near the receiver.

Multipath occurs when the signal arrives at the antenna from these reflected surfaces in addition to the line-of-sight source. The reflected signal is phase-shifted with respect to the original transmission and appears as additive noise at the antenna.

Multipath depends on satellite geometry and the antenna environment, which makes multipath difficult to model. For long observation periods — 24 hours or more — the multipath effects are partly reduced by averaging. However, observation periods usually last for only a few hours and often much less; this is why multipath is a problem.

Because the antenna locations are different, the multipath signa-

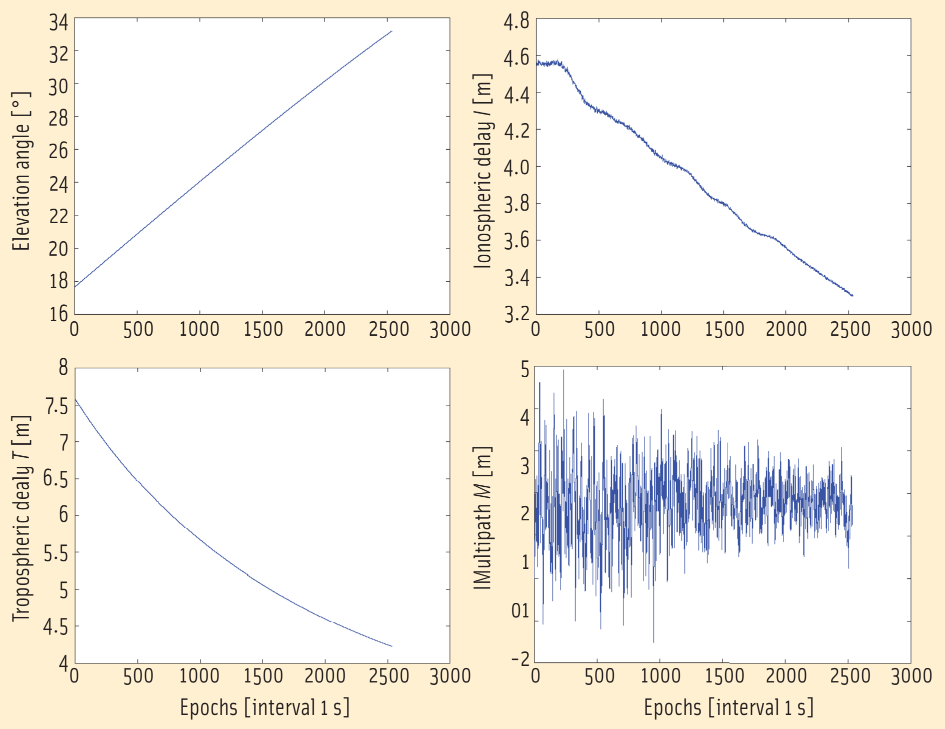


FIGURE 1 One-way errors for PRN14

ture at each antenna is unique and the error is not common mode.

Let the geometrical distance between satellite and receiver be ρ, let I be the ionospheric delay, and M the code multipath including receiver noise. The pseudoranges observed on L_1 and L_2 for PRN14 may then be expressed as

$$P_1 = \rho + I_1 + M_1 \quad (3)$$

$$P_2 = \rho + I_2 + M_2. \quad (4)$$

Remaining errors are identified as multipath and receiver noise — similarly for the phase observations:

$$\Phi_1 = \rho - I_1 + \lambda_1 N_1 + m_1 = \lambda_1 \varphi_1 \quad (5)$$

$$\Phi_2 = \rho - I_2 + \lambda_2 N_2 + m_2 = \lambda_2 \varphi_2. \quad (6)$$

The multipaths m_1 and m_2 on phase observations are so small that we subsequently put $m_i = 0$. We want to find an expression for M_1. We start by subtracting (5) from (3):

$$P_1 - \Phi_1 = 2I_1 + M_1 - \lambda_1 N_1$$

or

$$M_1 - \lambda_1 N_1 = P_1 - \Phi_1 - 2I_1 \quad (7)$$

and subtracting (6) from (5) yields

$$\Phi_1 - \Phi_2 = I_2 - I_1 + \lambda_1 N_1 - \lambda_2 N_2 = (\alpha - 1) I_1 + \lambda_1 N_1 - \lambda_2 N_2 \quad (8)$$

or

$$I_1 = \frac{1}{\alpha - 1}(\Phi_1 - \Phi_2) + \frac{1}{\alpha - 1}(\lambda_2 N_2 - \lambda_1 N_1). \quad (9)$$

We insert (9) into (7) and obtain

$$M_1 - \lambda_1 N_1 = P_1 - \Phi_1 - \frac{2}{\alpha - 1}(\Phi_1 - \Phi_2) - \frac{2}{\alpha - 1}(\lambda_2 N_2 - \lambda_1 N_1) \quad (10)$$

or

$$M_1 - \Big(\lambda_1 N_1 - \frac{2}{\alpha - 1}(\lambda_2 N_2 - \lambda_1 N_1)\Big) = P_1 - \Big(\frac{2}{\alpha - 1} + 1\Big)\Phi_1 + \frac{2}{\alpha - 1}\Phi_2. \quad (11)$$

We can reasonably assume that $E\{M_1\} = 0$. The second term on the left side of (11) is a constant; so, it is possible to reduce M_1 to M_1^* such that $E\{M_1^*\} = 0$:

$$M_1^* = P_1 - \frac{\alpha + 1}{\alpha - 1}\Phi_1 + \frac{2}{\alpha - 1}\Phi_2. \quad (12)$$

Analogously, by exchanging subscripts, we have for the multipath on L_2:

$$M_2^* = P_2 + \frac{\alpha + 1}{\alpha - 1}\Phi_2 - \frac{2}{\alpha - 1}\Phi_1. \quad (13)$$

For all epochs with all four observations P_1, P_2, Φ_1, and Φ_2 available, we compute multipath according to Equation (12). In the present case, the average error is below two meters and noisier at low satellite elevation angles. The noisy character of the plot reflects the actual noise in any pseudorange observation.

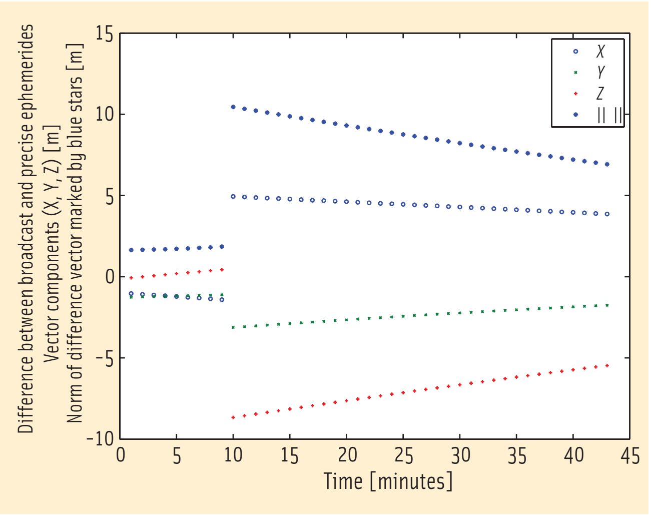


FIGURE 2 Difference in meters between precise and broadcast ephemerides for PRN14 for individual vector components (*X, Y, Z*). Norm of difference vector marked by stars

Finally we investigate the difference between precise and broadcast ephemerides. Any file containing broadcast ephemerides is specific to the site from which the satellites were tracked. Therefore, we need to identify the tracked satellites in the time period during which we want to compare the precise and broadcast ephemerides.

This can be done by inspecting the corresponding observation file in the PRN14 navigation message and getting the date and the seconds of the week; next we find the corresponding Julian Day Number (*jd*). We find an epoch (modulo 15 minutes) 60 minutes ahead of *jd*, which is counted in unit of day. (The variable *jd* is a real number where the integer part is number of whole days and the decimal part is made up of hours, minutes, seconds and fractions of seconds all convert into fractions of a day). Finally the GPS week number is computed by the Matlab function gps_time.

The precise ephemerides that we used can be found at <igscb.jpl.nasa.gov/igscb/product/1410>, the number 1410 being the GPS week number. The file extracts to igr14105.sp3, which contains precise ephemerides for January 19, 2007, in the SP3 format. The IGS SP3 files typically contain satellite positions (*X,Y,Z*) and clock offsets for each satellite every 15 minutes, from 0:00 hours to 23:45. (The SP3 file starts with some header lines, which for our purposes we can skip.)

We read precise orbits for the interpolation period plus three times 15 minutes ahead of the observation period and three times 15 minutes after the period, i.e., 13:15 hours, 13:30 hours, ... , 15:15 hours — in total nine sets. The precise coordinates for all tracked satellites are stored in the matrix *Xp*.

In the following discussion, we introduce a time scale with units of 15 minutes. We could have chosen whole minutes instead. The first observation is from 13.85 hours, and the last observation is taken at 14.55 hours. The entire observation period is 2,535 seconds = 42.25 minutes = 2.8 quarters of an hour.

A Lagrange interpolation, one for each coordinate, of at least seventh order is used to compute the actual position. The Matlab function intp does an (n – 1)th order polynomial Lagrange

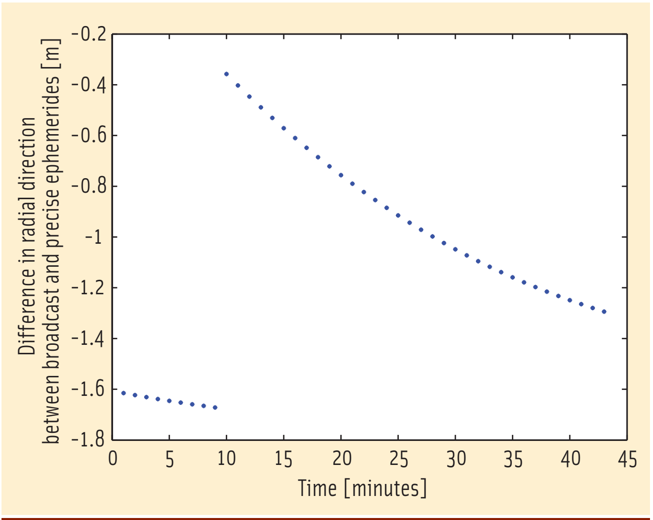


FIGURE 3 Length of difference vector between precise and broadcast ephemerides as projected onto the line of sight

interpolation. Let y be an $n \times 1$vector of data given at the discrete times x, (x is as well an $n \times 1$vector), (See the relevant discussion in *Matrix Analysis* by R. A. Horn and C. R. Johnson, Volume 1, pages 29–30, cited in Additional Resources at the end of this article.)

We want an interpolated precise position each minute; hence, we divide by 15 to keep units in quarters of an hour. The first parameter in the intp procedure describes the abscissae for the points at which we know the satellite coordinates. The second parameter contains these coordinates, and the third parameter describes the point set at which we want interpolated values — all in units of quarter of an hour. Each coordinate is interpolated separately!

Figure 2 shows the difference in satellite position as computed from broadcast ephemerides given via the navigation file, and the postprocessed satellite positions for PRN14. When comparing with broadcast ephemerides we assume the precise positions to be the true ones.

The actual influence of the orbit error on the receiver position is given by the projection of the difference vector onto the line between the receiver and satellite. Let vector $\boldsymbol{b}$ be the difference vector between the satellite positions computed from the broadcast and the precise ephemerides, and $\boldsymbol{a}$ to be the vector between the receiver and the satellite position as computed from broadcast ephemerides. Then the projected difference vector $\boldsymbol{p}$ onto vector $\boldsymbol{a}$ is given as

$$\boldsymbol{p} = \frac{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}}{\boldsymbol{a}^{\mathrm{T}}\boldsymbol{a}}\boldsymbol{a}.$$

Figure 3 shows vector $\boldsymbol{p}$ for PRN14. The computed ephemeris error varies in the range of ±2 meters. The discontinuities in the error reflect the routine ephemeris updates at two-hour intervals.

Additional Resources

Horn, R. A., and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, Cambridge, 1985

Author

Originally graduated as a chartered surveyor, **Kai Borre** obtained his Ph.D. in geodesy from Copenhagen University, and a Doctor of Technology degree from Graz University of Technology. He has been a full professor in geodesy at Aalborg University, Denmark, since 1976.

For more than 30 years Borre has conducted research and taught in the area of satellite-based positioning. In 1996 he established the Danish GPS Center and since 2000 has been head of a two-year international M.Sc. program in GPS technology.

Borre is a coauthor of the widely used textbooks, *Linear Algebra, Geodesy, and GPS* and *A Software-Defined GPS and Galileo Receiver; Single-Frequency Approach*.

Since the early 1990s, Borre has published Matlab code for processing of GPS observations. Since 2003 he has published Matlab code together with explanatory text, a successful new pedagogical concept. IG