Accepted for publication in GPS Tool Box, GPS Solutions, 13 Apr 2011

SATLSim: A Semi-Analytic Framework for Fast GNSS Tracking Loop Simulations

Daniele Borio, Pratibha B. Anantharamu and Gérard Lachapelle

PLAN Group, Department of Geomatics Engineering, University of Calgary
Phone: +1 403 220 9797
E-mail addresses:
daniele.borio@ieee.org, pbananth@ucalgary.ca, lachapel@ucalgary.ca

Introduction

Digital tracking loops are nonlinear in nature and their complexity has increased with the advent of new Global Navigation Satellite Systems (GNSS) signals such as the Binary Offset Carrier (BOC) modulation. For these reasons, their analysis is often confined to Monte Carlo simulations that provide enough flexibility to fully characterize the system under analysis. This flexibility is, however, achieved at the expense of long simulation times or it can lead to the use of a limited number of simulation runs. In order to overcome these limitations, alternative solutions have been explored (Jeruchim et al. 2000, Tranter et al. 2004). In this respect, Semi-Analytic techniques (Tranter et al. 2004) exploit the knowledge of the system under analysis to reduce the computational load that full Monte Carlo simulations would require. In the Semi-Analytic approach, only nonlinear blocks are fully simulated whereas analytical results are used to account for the linear components of the system. This principle directly applies to GNSS digital tracking loops, where the most computationally demanding components are the Integrate and Dump (I&D) blocks used for despreading the incoming GNSS signals. These blocks are linear and accurate analytical models can be used for their characterizations. Thus, Semi-Analytic simulations, exploiting an analytical model for the I&D blocks, can be developed (Golshan 2006, Silva et al. 2007, Borio et al. 2010).

The Semi-Analytic framework developed in Borio et al. (2010) has led to the development of the Matlab® Semi-Analytic Tracking Loop Simulations (SATLSim) toolbox. This framework is briefly discussed and the Matlab® code for the fast simulation of GNSS digital tracking loops is presented.

SATLSim has been organized in a modular way where each fundamental block for the tracking loop simulation is implemented in separate functions. By modifying these functions it is possible to simulate different tracking loops, including delay, frequency and phase lock loops (DLL, FLL and PLL). The simulation scheme is flexible enough to allow the analysis of unambiguous BOC tracking algorithms (Borio et al. 2010) and the specific case of the Double Estimator (Hodgart and Blunt 2007) is considered.

Semi-Analytic Simulation Scheme

In a GNSS tracking loop, the incoming signal is correlated with several locally generated replicas and different correlator outputs are produced. This operation is performed by the I&D blocks and each correlator output is a function of the input signal and the parameters previously estimated by the tracking loop. The correlator outputs are passed to the nonlinear discriminator that produces a first estimate of the tracking error that the loop is trying to minimize. The tracking error is filtered and passed to the Numerically Controlled Oscillator (NCO) that is used for generating new local signal replicas.
Efficient tracking loop simulations can be obtained by substituting the I&D blocks with their analytical model. More specifically, a correlator output can be modeled as:

$$\sqrt{\frac{C}{2}}\frac{\sin\left(\pi\Delta f_d T_c\right)}{\pi\Delta f_d T_c}R_l\left(\Delta\tau_d,\Delta\tau_s\right)\exp\left\{j\Delta\phi\right\}+\eta_c \qquad (1)$$

where

- C is the received signal power;
- Δf_d and $\Delta\phi$ are the residual frequency and phase errors;
- $\Delta\tau_d$ and $\Delta\tau_s$ are the code and subcarrier delay errors. It is noted that, in a standard tracking loop architecture, local code and subcarrier are aligned and $\Delta\tau_s = 0$. $\Delta\tau_s$ can be different from zero only when a Subcarrier Lock Loop (SLL) is used to correctly align the signal subcarrier (Hodgart and Blunt 2007);
- T_c is the coherent integration time;
- $R_l\left(\Delta\tau_d,\Delta\tau_s\right)$ is the correlation function between incoming and locally generated code and is a function of both code and subcarrier delay errors.

When $\Delta\tau_s = 0$, $R_l\left(\Delta\tau_d, \Delta\tau_s\right)$ degenerates to the standard code correlation function;

- η_c is a zero mean noise term whose variance depends on the input noise power, front-end filtering and the correlation process operated by the I&D blocks. More details on the properties of η_c can be found in Borio et al. (2010).

From (1), it is possible to reconstruct the correlator outputs given the estimation errors generated by the loop. Thus, the correlation process does not need to be simulated and only the estimation errors are determined using a Monte Carlo approach. Based on this principle, the simulation scheme shown in Figure 1 can be adopted for the fast simulation of digital tracking loops.

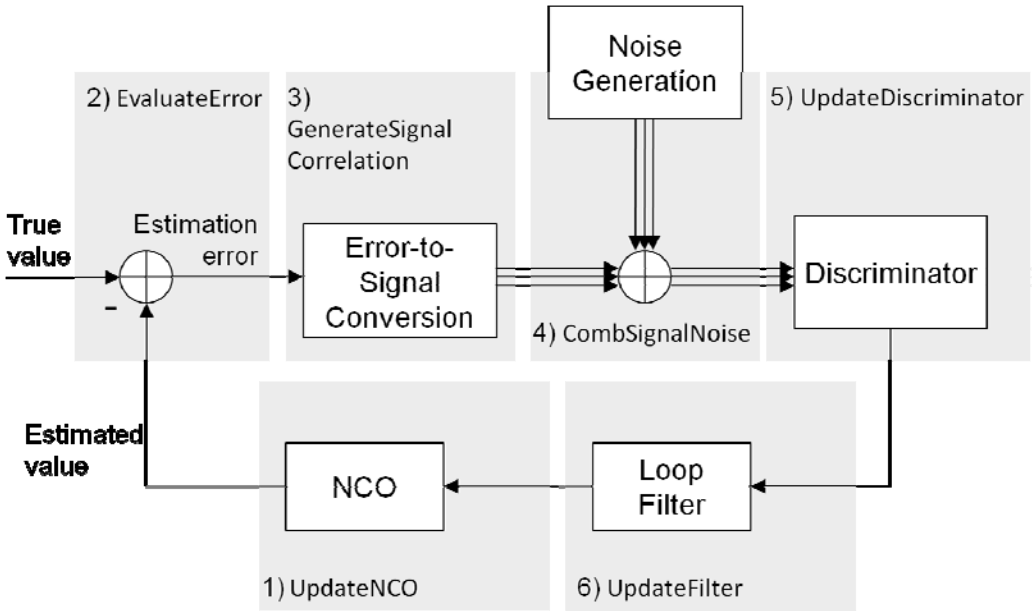

Figure 1: Semi-Analytic scheme adopted for the simulation of GNSS tracking loops. The name associated to each functional block corresponds to the Matlab® function implemented in the SATLSim toolbox.

The functional blocks in Figure 1 are numbered according to the execution order in the provided code and names associated to each block correspond to different Matlab® functions implemented in the SATLSim toolbox. By modifying these functional blocks, it is possible to simulate different tracking loops.

In the proposed simulation scheme, a new estimate of the tracking parameters (Doppler frequency, carrier phase and code and subcarrier delays) is generated by an NCO model. This model accounts for the integration process performed by a real NCO and different update equations can be used (Stephens and Thomas

1995). A commonly used model is the rate-only feedback NCO (Stephens and Thomas 1995), characterized by the following update equation:

$$\hat{\varphi}_k = \hat{\varphi}_{k-1} + \frac{T_c}{2}\left(\delta\hat{\varphi}_{k-1} + \delta\hat{\varphi}_{k-2}\right) \quad (2)$$

where $\hat{\varphi}_k$ denotes the *k*-th estimate of the tracking parameter under consideration and $\delta\hat{\varphi}_k$ is its estimated rate of change. $\delta\hat{\varphi}_k$ is generally provided by the loop filter. It is noted that when several parameters are considered, equation (2) is used to update each term independently. The new parameter estimate is compared against the true value and a new estimation error is computed. This error is then used for the generation of the signal component at the output of the I&D block using equation (1). The noise term, generated separately, is then added to the signal component. When several correlators are required, the correlation among the different noise components has to be accounted for. This is simulated using the approach described in Borio et al. (2010).

The operations required to convert the correlator outputs into a new estimate of the parameter rate, $\delta\varphi_k$, are fully simulated and correspond to the functional blocks that can be found in a real tracking loop. For instance, the correlator outputs can be used to update the nonlinear discriminator and the loop filter. It is noted that a similar simulation scheme can be used for analyzing Kalman filter based tracking. In this case, the correlator outputs are fed to a Kalman filter that is used to produce new estimates of the tracking parameters.

Code Structure

The structure of the code for the Semi-Analytic simulation scheme is provided in Figure 2. In this case, the code is used to estimate the tracking jitter of the loop as a function of different parameters, such as the Early-minus-Late spacing and the input Carrier-to-Noise density power ratio (C/N_0).

The parameters required for initializing the simulation procedure are accessible through the function `InitSettings`. These parameters include the sampling frequency, the loop bandwidth and the coherent integration time that are used to design the loop filters through the function `FilterDesign`. In the provided code, standard formulae from Kaplan and Hegarty (2006) are used. However, `FilterDesign` can be modified in order to adopt a different approach, such as

the controlled-root formulation proposed by Stephens and Thomas (1995). During the initialization phase, the true input parameters are also generated.
The simulation core consists of three nested loops, on the Early-minus-Late spacing, for different C / N_0 values and for the number of simulation runs.

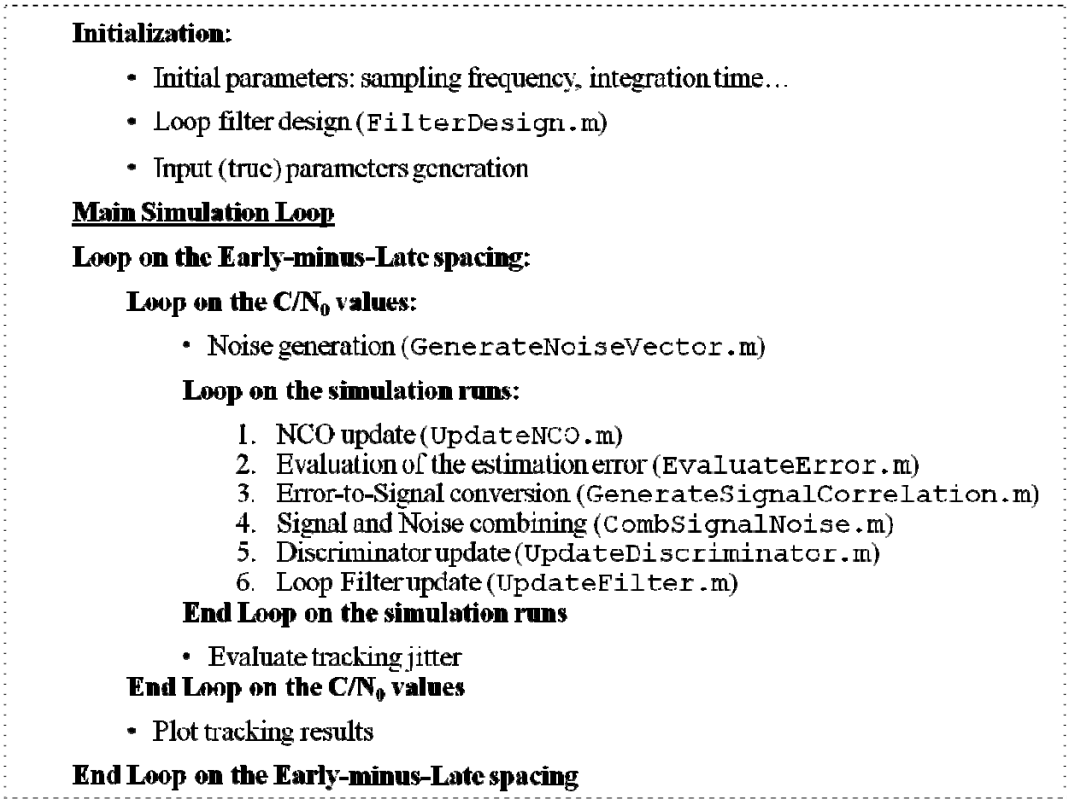

Figure 2: Code structure and different functional blocks.

It is noted that the loop on Early-minus-Late spacing can be absent if, for example, only a PLL is considered. For each Early-minus-Late spacing and for a fixed C / N_0, a noise vector containing the noise components of the correlator outputs is generated. The vector length is equal to the number of simulation runs and all the noise components are generated at once for efficiency reasons.
After generating the noise components, the inner loop on the simulation runs starts. In this loop, the operations described in the previous section are performed. Each functional block is implemented by a different Matlab® function, as indicated in Figure 2.
All intermediate results, such as the discriminator and loop filter outputs, are stored in auxiliary vectors and are used at the end of the loop on the simulation runs to evaluate quantities of interest such as the tracking jitter.

In the provided code, theoretical formulae for the computation of the tracking jitter are also implemented and used as a comparison term for the simulation results.

Standard PLL (PLL.m)

The simulation of a standard PLL requires the generation of the Prompt correlator alone (`GenerateSignalCorrelation`). The Prompt correlator is the output of I&D block computed with respect to the best delay estimate provided by the loop (Kaplan and Hegarty 2006). For this reason, the noise generation (`GenerateNoiseVector`) simply consists of simulating a one dimensional complex Gaussian white sequence with independent and identically distributed real and imaginary parts with variance (Borio et al. 2010)

$$\sigma_n^2 = \frac{1}{2C / N_0 T_c}. \tag{3}$$

When simulating a standard PLL alone, perfect code synchronization is assumed and (1) simplifies to

$$\sqrt{\frac{C}{2}} \frac{\sin\left(\pi \Delta f_d T_c\right)}{\pi \Delta f_d T_c} \exp\left\{ j \Delta\phi \right\} + \eta_c \tag{4}$$

where Δf_d is obtained by comparing the true Doppler frequency against the loop filter output. $\Delta\phi$ is the phase error obtained as the difference between the true phase and the phase estimate produced by the NCO.
In the provided code, the function `UpdateDiscriminator` implements a standard Costas discriminator. Different phase discriminators, as indicated in Kaplan and Hegarty (2006), can be easily implemented by changing this function.

Double Estimator (DoubleEstimator.m)

In the Double Estimator case, the function `GenerateNoiseVector`, responsible for the generation of the correlator noise, produces a $5 \times N$ matrix, where N is the number of simulation runs. The five rows of this matrix correspond to the five correlators required by the Double Estimator that are characterized by the following correlation matrix

$$C_n = \begin{bmatrix} 1 & R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) & R_l\left(\frac{d_s}{2},0\right) & R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) & R_l\left(d_s,0\right) \\ R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) & 1 & R_l\left(0,\frac{d_{sc}}{2}\right) & R_l\left(0,d_{sc}\right) & R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) \\ R_l\left(\frac{d_s}{2},0\right) & R_l\left(0,\frac{d_{sc}}{2}\right) & 1 & R_l\left(0,\frac{d_{sc}}{2}\right) & R_l\left(\frac{d_s}{2},0\right) \\ R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) & R_l\left(0,d_{sc}\right) & R_l\left(0,\frac{d_{sc}}{2}\right) & 1 & R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) \\ R_l\left(d_s,0\right) & R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) & R_l\left(\frac{d_s}{2},0\right) & R_l\left(\frac{d_s}{2},\frac{d_{sc}}{2}\right) & 1 \end{bmatrix} \quad (5)$$

where d_s and d_{sc} are the code and subcarrier Early-minus-Late spacing.

The NCO update (`UpdateNCO`) is performed on both code and subcarrier loops and the estimated errors, $\Delta\tau_d$ and $\Delta\tau_s$, are used to compute new correlator signal components (`GenerateSignalCorrelation`). Two nonlinear discriminators (`UpdateDiscriminator`) and loop filters (`UpdateFilter`) are run in parallel to determine the rate of change of both code and subcarrier delay.

The Double Estimator provides an example of how several tracking loops operating in parallel can be easily coupled in order to provide more realistic simulations accounting for the interaction of different tracking algorithms (Borio et al. 2010).

Installation

The SATLSim toolbox can be found on the GPS Toolbox web site at http://www.ngs.noaa.gov/gps-toolbox/. The latest version can also be obtained from the Position Location And Navigation (PLAN) website under the section Publications (http://plan.geomatics.ucalgary.ca/publications.php). To install the toolbox, download the SATLSim.zip file and unzip it. Ensure the SATLSim directory location is specified in the Matlab search path.

To evaluate the performance of the Double Estimator, follow the instructions provided in the SATLSim\DoubleEstimator\README.txt file. To evaluate the performance of the standard PLL, follow the instructions provided in the SATLSim\PLL\README.txt file.

The toolbox has been designed as a stand-alone application and no additional packages are required other than the functions already available in a standard Matlab® installation.

Sample Results

Sample results, obtained using the proposed Semi-Analytic framework (PLL.m), are shown in Figure 3, where the tracking jitter for a second order PLL is provided. The loop filter has been designed using the approach described in Kaplan and Hegarty (2006) that is based on the bilinear transform and on results derived from analog PLL theory. When the product between loop bandwidth and coherent integration time is significantly greater than zero, this method fails in providing a PLL matching the design parameters.

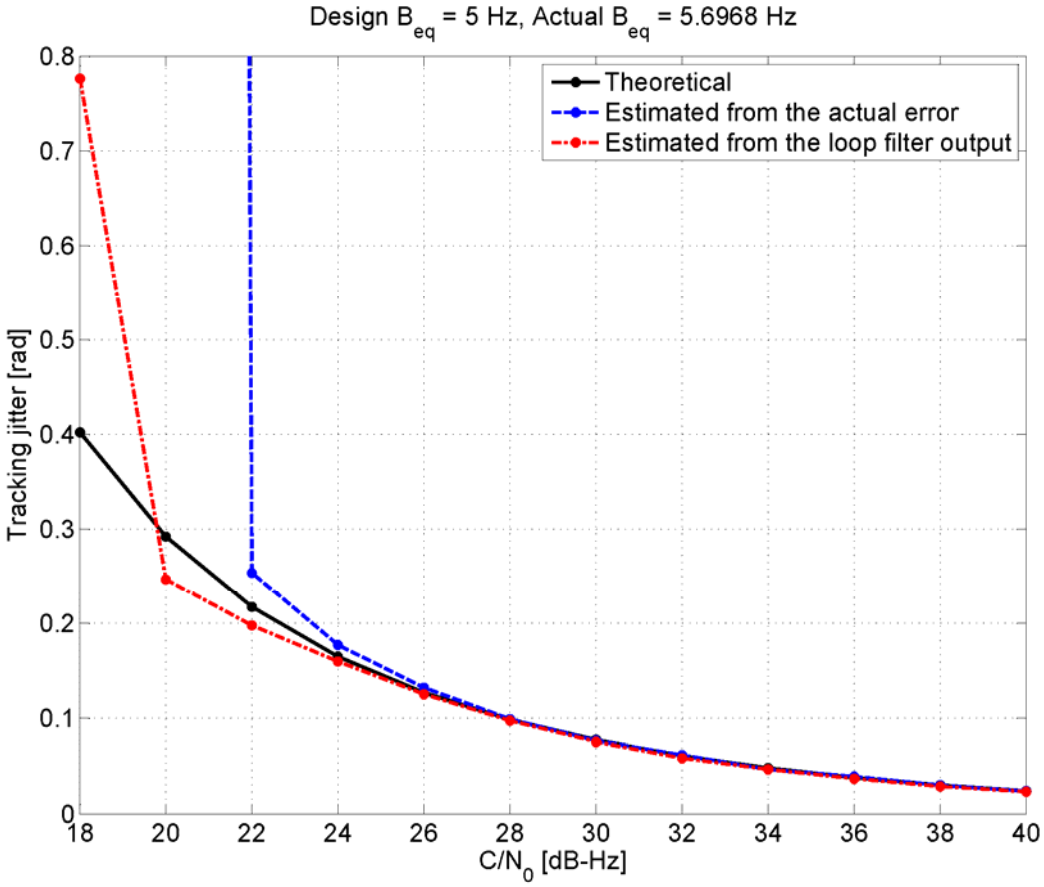

Figure 3: Tracking jitter for a second order PLL as a function of the C/N_0. Coherent integration time equal to 10 ms.

This fact is highlighted in Figure 3, where both design and actual bandwidths are indicated. When the product between loop bandwidth and integration time is equal to 0.05, it is already possible to observe a divergence between design and actual

loop bandwidths. The actual bandwidth is computed by the function `ComputeEquivBW`.

In Figure 3, three curves are present. The first one, indicated by "Theoretical", is obtained using the theoretical expression from Kaplan and Hegarty (2006) and the actual loop bandwidth. The second one is obtained by estimating the standard deviation of the phase tracking error that is computed during the second step of the proposed Semi-Analytic methodology. In the last curve, the tracking jitter is derived by opportunely propagating the standard deviation of the loop filter output. More specifically, in the PLL linear approximation, the tracking error and the loop filter output are related by a linear relationship. Thus, the standard deviation of the loop filter output and the tracking jitter are related by a constant factor. This propagation factor is evaluated in the provided code by the function `NoisePropagation`. The vertical trend in the curve computed from the actual tracking error indicates that the loop has lost lock. This phenomenon is not predicted by the theoretical jitter model based on a linear approximation of the loop.

The results shown in Figure 3 were obtained using the `PLL.m` function in the SATLSim toolbox. Similar results are obtained in the case of the Double Estimator (Borio et al. 2010) using the `DoubleEstimator.m` function.

Conclusions

A Semi-Analytic framework for the fast simulation of digital tracking loops has been presented. Although the code provided has been specifically designed for evaluating the tracking jitter, the proposed framework can be used for quantifying other figures of merit including tracking threshold and mean time to lose lock. The proposed framework is general and can be used for the simulation of new algorithms such as unambiguous BOC tracking (Borio et al. 2010) and the analysis of collaborative code tracking techniques (Borio et al. 2009).

References

Borio D., Anantharamu P. B. and Lachapelle G. (2010) "Semi-Analytic Simulations: An Extension to Unambiguous BOC Tracking," in Proceedings of ION International Technical Meeting (ITM), San Diego, 14 pages, January.

Borio D., Mongrédien C. and Lachapelle G. (2009) “Collaborative Code Tracking of Composite GNSS Signals”, IEEE Journal of Selected Topics in Signal Processing, Vol. 3, No. 4, pp. 613-626, July.

Golshan A. R.(2006) “Post-Correlator Modeling for Fast Simulation and joint Performance Analysis of GNSS Code and Carrier Tracking Loops,” in Proc. of the ION National Technical Meeting (NTM), Monterey, CA, pp. 312 –318, January.

Hodgart M.S. and Blunt P. D. (2007) “A Dual Estimate Receiver of Binary Offset Carrier (BOC) Modulated Signals Global Navigation Satellite Systems,” Electronics Letters, Vol. 43, No. 16, pp. 877-878, August.

Jeruchim M. C., Balaban P. and Shanmugan K. S. (2000) “Simulation of Communication Systems,” 2nd ed. Springer, October.

Kaplan E. D. and Hegarty C. J. (2006), Understanding GPS: Principles and Applications, 2nd ed. Norwood, MA, USA: Artech House Publishers

Silva J. S., Silva P. F., Fernández A., Diez J.and Lorga J. F. M. (2007) “Factored Correlator Model: a Solution for Fast, Flexible, and realistic GNSS Receiver Simulations,” in Proceedings of ION/GNSS, Forth Worth, TX, US, pp. 2676 – 2686, September.

Stephens S. A. and Thomas J. B. (1995) “Controlled-Root Formulation for Digital Phase-Locked Loops,” IEEE Transactions on Aerospace and Electronic Systems, Vol. 13, No. 1, pp. 78-95, January.

Tranter W. H., Shanmugan K. S., Rappaport T. S. and Kosbar K.L. (2004), “Principles of Communication Systems Simulation with Wireless Applications”, Prentice Hall, Communications Engineering and Emerging Technologies Series, January.

GPS Tool Box

The GPS Tool Box is a column dedicated to highlighting algorithms and source code utilized by GPS engineers and scientists. If you have an interesting program or software package you would like to share with our readers, please pass it along; e-mail it to us at gps-toolbox@ngs.noaa.gov. To comment on any of the source code discussed here, or to download source code, visit our website at http://www.ngs.noaa.gov/gps-toolbox. This column is edited by Stephen Hilla, National Geodetic Survey, NOAA, Silver Spring, Maryland, and Mike Craymer, Geodetic Survey Division, Natural Resources Canada, Ottawa, Ontario, Canada