Smoother-Based GPS Signal Tracking in a Software Receiver

Mark L. Psiaki, *Cornell University*

BIOGRAPHY

Mark L. Psiaki is an Associate Professor of Mechanical and Aerospace Engineering at Cornell University. His research interests are in the areas of estimation and filtering, spacecraft attitude and orbit determination, and GPS technology and applications. He holds a B.A. in Physics and M.A. and Ph.D. degrees in Mechanical and Aerospace Engineering, all from Princeton University.

ABSTRACT

Global Positioning System (GPS) signal tracking algorithms have been developed using the concepts of Kalman filtering and smoothing. The goal is to improve phase estimation accuracy for non-real-time applications. A bit-grabber/software-receiver has been developed for the GPS *L1* coarse/acquisition signal. The bit grabber down-converts, digitizes, and stores the raw RF signal. The software receiver tracks each signal using a 2-step process. The first step uses phase-locked and delay-locked loops. The second step refines the tracking accuracy through the use of linear smoothing techniques. These techniques make optimal use of after-the-fact data.

INTRODUCTION

A GPS user receiver needs to track the spread-spectrum signals that it receives from the GPS constellation. Almost all receivers track the phase of the pseudo-random number (PRN) code that is used to spread the signal's spectrum, and many receivers also track the phase of the underlying carrier signal. The phase of the PRN code is used to infer the pseudo range from the GPS satellite to the user, and the accuracy with which this phase can be tracked constitutes a fundamental limit to the achievable accuracy of the receiver's determination of absolute position and time. The phase of the carrier signal is used to determine the Doppler shift and the accumulated delta range. Accurate carrier phase tracking is necessary for precise differential GPS measurements and for precise velocity determination.

Standard receivers track code phase using a Delay-Locked Loop (DLL) and carrier phase using a Phase-Locked Loop (PLL) [1,2]. These entities are feedback loops

that align a replica signal in the receiver with the actual received signal. In a normal receiver, these loops must operate in real-time, which means that they can rely only on past measurements of phase errors in order to align the two signals. This causality constraint limits the receiver's ability to accurately measure code and carrier phase.

Recently there has been a lot of interest in the use of software receivers in conjunction with bit-grabbers [3-11]. A bit-grabber samples a down-converted and filtered version of the raw GPS radio frequency (RF) signal and either stores it on disk, sends it to a telemetry system, or sends it directly to a microprocessor. The remainder of the receiver functions are implemented in software in a microprocessor, hence the name "software receiver." These include base-band carrier mixing, PRN code correlation, and signal tracking. A typical real-time receiver implements these functions mostly in dedicated digital hardware because they involve a large computational load. Implementation in a software receiver poses execution speed challenges if the system must act in real-time [11].

There are a number of applications in which non-real-time data processing is useful. One is to determine what happened during an interval when a real-time receiver lost lock [7,8]. After-the-fact processing can help sort out the cause because loss of lock is not an issue for bit-grabbed raw RF data. Another use for after-the-fact GPS data processing is in the acquisition of navigation information from a signal of limited duration [4,6,10]; a single short data interval can be used both for acquisition and to derive phase observables. Software post-processing can be useful when there is an extremely low signal-to-noise ratio (SNR) because it can incorporate sophisticated algorithms that allow the use of longer averaging intervals [10]. After-the-fact software receivers can be used in flight testing of small payloads. A test vehicle can be equipped with a bit grabber and a telemetry system that sends the raw data bits to a ground station. The ground station can then process the data for use in analysis of the flight test.

This paper concentrates on the design of new signal tracking functions for use in non-real-time processing of raw GPS intermediate frequency (IF) signals. Its goal is to improve the accuracy of a receiver's estimates of the

PRN code phase and carrier phase. It does this by using data that extends into the "future" – that is, beyond the time point of interest. It employs an algorithm called a smoother, which is a variant of a Kalman filter [12]. Note, that the terms "smoother" and "smoothing," as used in this paper, do not refer to the concept that is commonly known in the GPS literature as carrier-aided smoothing.

Reference 5 and related works by the same authors present the only published signal tracking algorithm that has been designed specifically for use in a GPS software receiver. This algorithm uses the discrete Fourier transform as part of its code correlation process, and it uses simple feedback principles to effectively implement a frequency-locked carrier tracking loop and a delay-locked loop that tracks the code phase.

Two works that are more relevant to this research come from the general area of real-time GPS signal tracking [13,14]. These works use Kalman filtering theory in order to design phase-locked loops for tracking the GPS carrier signal. Their signal models and Kalman filter design techniques can be used and extended in order to develop smoothers to track both carrier phase and code phase.

The present paper makes 3 contributions. First, it develops signal models for the code and carrier phase that are suitable for the purpose of designing smoothers. Second, it presents smoother designs that act on bit-grabbed data and optimally estimate carrier phase and code phase. This represents the paper's primary contribution and is the first use of smoothers in the field of GPS RF signal tracking. Third, the paper tests the smoothers using real GPS data that has been collected by a bit-grabber receiver and a roof-mounted antenna.

These contributions yield an ability to track GPS code phase and carrier phase with a smaller level of receiver-induced error. This increased accuracy can be significant in differential GPS applications, in situations with a very low SNR, or in cases where the user vehicle is undergoing highly dynamic maneuvers.

This paper represents a first cut at the application of smoothing algorithms to GPS signal tracking. Its goals are to explain the general technique and to show how the simplest possible smoothers can yield improvements. Additional work will be needed in order to realize the fullest possible benefits of smoothing techniques.

The techniques of this paper are generally applicable to both the coarse/acquisition (C/A) code on the *L1* frequency and to the precision (P) code on both the *L1* and *L2* frequencies. It is necessary to know the code in order to implement this paper's methods. This paper targets its developments to the C/A code because the anti-spoofing Y encryption of the P code precludes civilian testing of these concepts on P code.

The remainder of this paper is divided into 6 sections plus conclusions. The second section describes the hardware and functions of the bit-grabber/software-receiver system. The third section presents mathematical models for the dynamic evolution of the carrier phase and the code phase. The fourth section explains how to design Kalman filters for purposes of phase tracking. Kalman filters provide a basis for understanding smoothers. In addition, they are used in the first step of a two-step signal tracking process. The fifth section designs the smoothers that carry out the carrier phase and code phase tracking. The results of signal tracking experiments are presented in the sixth section. The seventh section suggests enhancements that could be made to the smoothing algorithms.

HARDWARE AND FUNCTIONAL DESCRIPTION OF A BIT-GRABBER/SOFTWARE-RECEIVER SYSTEM

This paper's signal tracking algorithms function within the framework of a software receiver that operates on data from a GPS bit grabber. The overall bit-grabber/software-receiver system is depicted schematically in Fig. 1. The bit grabber is a special-purpose electronics card that down-converts, band-pass filters, and gain adjusts the *L1* RF signal $y_{L1}(t)$. The result is an intermediate-frequency RF signal, $y_{IF}(t)$. This latter signal gets digitized and sampled by an analog-to-digital converter (ADC). This sampled signal is stored on a computer hard drive for later post-processing by the software receiver. The software receiver reads $y_{L1}(t)$ from the disk and processes it in order to acquire and track any GPS signals that are present in it.

The performance of this system is relatively insensitive to the specific characteristics of the bit grabber, but for the sake of completeness, the hardware that has been used in this study is now described. The RF front end is a Plessey GP2015 chip. It has 3 stages of mixing, 3 stages of band-pass filtering, and an automatic gain control loop. Its output maps the nominal *L1* carrier frequency to an intermediate frequency of 4.309 MHz, and the signal is filtered to a half bandwidth is 1 MHz. This signal is sampled by a 2-bit ADC at a sampling frequency of 5.714 MHz. This aliases the nominal intermediate frequency of the sampled signal to 1.405 MHz, and it causes a phase reversal. The RF front end, the ADC, and the sampler are all implemented on a single chip [15]. The bit-grabber uses a temperature-compensated crystal oscillator as its timing reference. Its one-second root Allan variance is no greater than 10^{-9} (see Ref. 16).

The bit-grabber's effect on a GPS *L1* C/A signal can be modeled mathematically. Suppose that the signal from a single GPS satellite comes out of the antenna in the form:

$$y_{L1}(t) = A\,C(t)\,D(t)\cos[\omega_{L1}t + \phi(t)] \qquad (1)$$

In this formula *A* is the signal's amplitude, $C(t)$ is the pseudo-random spreading code (± 1 with a 1.023 MHz

chipping rate), and $D(t)$ is the encoded data bit of the navigation message (± 1 with a 50 Hz data bit rate) [17]. The frequency $\omega_{L1} = 1575.42 \times 10^6 \times 2\pi$ rad/sec is the nominal $L1$ carrier frequency, and $\phi(t)$ is the carrier phase perturbation due to the integrated Doppler shift. The bit grabber operates on the signal in eq. (1) to produce the following down-converted signal at the output of its ADC:

$$y_{IF}(t_j) = B\, C(t_j)\, D(t_j) \cos[\omega_{IF} t_j - \phi(t_j)] + v_{d(j)} \qquad (2)$$

where t_j is the sample time, B is the output amplitude, ω_{IF} is the down-converted image of the $L1$ carrier frequency, and $v_{d(j)}$ is digitization error. $\omega_{IF} = (88.54/63) \times 10^6 \times 2\pi$ rad/sec for the implementation that has been used in this study. The sign in front of $\phi(t_j)$ is reversed in eq. (2) from what it is in eq.(1). This phase reversal is the result of the aliasing that occurs during the 5.714 MHz sampling process. Note that eq. (2) neglects distortion and delay that are caused by the RF front end's band-pass filters. The distortion affects the shapes of $C(t)$ and $D(t)$, and a common delay applies to $C(t)$, $D(t)$ and $\phi(t)$. Neglect of these effects is reasonable. The distortion is not very large, and the delay can be treated as an additive receiver clock error.
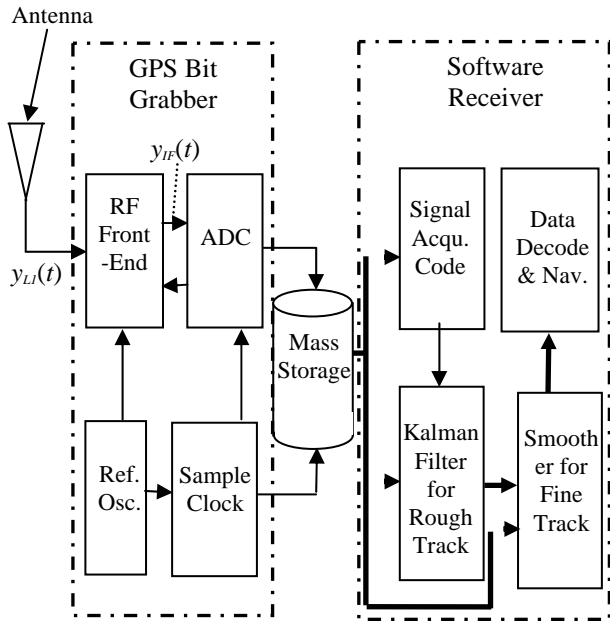

Fig. 1. Schematic block diagram of a GPS bit-grabber/software-receiver system.

The software receiver portion of this system includes 4 basic signal processing functions. The signal acquisition section generates initial estimates of the code phase and Doppler shift of a given signal. The Kalman filter section implements first-cut tracking of the code and carrier of each signal and operates much a conventional DLL/PLL channel. The smoother block uses the outputs of the Kalman filter block and the raw $y_{IF}(t)$ data to generate refined estimates of each signal's code and carrier phase.

The Data decoding and navigation block includes the software that decodes each signal's navigation message and that uses the results of the signal tracking blocks to deduce pseudo-range and the navigation solution.

The signal acquisition process searches for the C/A PRN code start time, $\hat{T}_k$, and the Doppler shift, $d\phi/dt$, of the signal from a given GPS satellite. The search computes the cross correlation between $y_{IF}(t_j)$ and a replica that includes the PRN code and the carrier signal. It surveys the 2-dimensional ($\hat{T}_k$, $d\phi/dt$) space in order to find a strong cross-correlation peak. This process uses a Fourier-transform-based approach that simultaneously computes the correlation for all code delays of interest at a given Doppler shift [5]. For strong signals, the search computes its correlations using a single millisecond's worth of data from the bit grabber, which equals one period of the C/A PRN code. For weaker signals, it uses several C/A code periods. The subject of signal acquisition in a software receiver has been treated by other researchers, e.g., see Ref. 5, and the present paper merely makes use of existing results.

The Kalman Filter and Smoother modules in the software receiver implement functions like those of the DLLs and PLLs of a conventional real-time receiver: They estimate the phases of the code and the carrier. They also estimate the frequency and drift rate of the carrier. The main difference from a conventional receiver is that the smoother block uses correlations which extend past the time point of interest. These two blocks are the subjects of the remainder of this paper.

MODELS OF CARRIER-PHASE AND CODE-PHASE MEASUREMENTS AND DYNAMICS

Correlation-Based Phase Measurements

The measurement process begins with reconstructions of the code phase and the carrier phase of the signal. The reconstructed code phase is stored in terms of estimated start/stop times of the C/A PRN code periods, $\bar{T}_0$, $\bar{T}_1$, $\bar{T}_2$, ..., $\bar{T}_{k-1}$, $\bar{T}_k$, $\bar{T}_{k+1}$, ... Suppose that $C_0(t)$ is the nominal PRN code for the tracked satellite. It is a function with values of ± 1, and its period starts at $t = 0$ and lasts 0.001 sec. The estimated start-stop times are used to reconstruct the received PRN code according to the following formula:

$$\bar{C}(t) =
\begin{cases}
\vdots & \vdots \\
C_0[0.001(t - \bar{T}_{k-1})/(\bar{T}_k - \bar{T}_{k-1})] & \text{if } \bar{T}_{k-1} \le t < \bar{T}_k \\
C_0[0.001(t - \bar{T}_k)/(\bar{T}_{k+1} - \bar{T}_k)] & \text{if } \bar{T}_k \le t < \bar{T}_{k+1} \\
\vdots & \vdots
\end{cases} \qquad (3)$$

The reconstructed carrier signal is based on linear

interpolation between reconstructed carrier phases at the estimated code period start/stop times. Suppose that $\phi_{re(k)}$ is the reconstructed carrier phase perturbation at time $\overline{T}_k$. Then the following two signals are, respectively, the in-phase and quadrature reconstructions of the IF image of the carrier signal:

$$\overline{y}_I(t) =$$
$$\begin{cases} \vdots & \vdots \\ \cos[\omega_{IF}t - \phi_{re(k-1)} - \omega_{re(k-1)}(t - \overline{T}_{k-1})] & \text{if } \overline{T}_{k-1} \leq t < \overline{T}_k \\ \cos[\omega_{IF}t - \phi_{re(k)} - \omega_{re(k)}(t - \overline{T}_k)] & \text{if } \overline{T}_k \leq t < \overline{T}_{k+1} \\ \vdots & \vdots \end{cases}$$

(4a)

$$\overline{y}_Q(t) =$$
$$\begin{cases} \vdots & \vdots \\ -\sin[\omega_{IF}t - \phi_{re(k-1)} - \omega_{re(k-1)}(t - \overline{T}_{k-1})] & \text{if } \overline{T}_{k-1} \leq t < \overline{T}_k \\ -\sin[\omega_{IF}t - \phi_{re(k)} - \omega_{re(k)}(t - \overline{T}_k)] & \text{if } \overline{T}_k \leq t < \overline{T}_{k+1} \\ \vdots & \vdots \end{cases}$$

(4b)

where $\omega_{re(k)} = [\phi_{re(k+1)} - \phi_{re(k)}]/[\overline{T}_{k+1} - \overline{T}_k]$ is the reconstructed Doppler shift on the time interval from $\overline{T}_k$ to $\overline{T}_{k+1}$.

The reconstructed signals in eqs. (3)-(4b) can be used to measure carrier phase and code phase errors. The phase error measurements make use of 4 correlations:

$$I_{e(k)} = \sum_{j=j_{start(k)}}^{j_{stop(k)}} y_{IF}(t_j)\,\overline{y}_I(t_j)\,\overline{C}(t_j + 0.5\Delta t_{eml}) \quad (5a)$$

$$Q_{e(k)} = \sum_{j=j_{start(k)}}^{j_{stop(k)}} y_{IF}(t_j)\,\overline{y}_Q(t_j)\,\overline{C}(t_j + 0.5\Delta t_{eml}) \quad (5b)$$

$$I_{l(k)} = \sum_{j=j_{start(k)}}^{j_{stop(k)}} y_{IF}(t_j)\,\overline{y}_I(t_j)\,\overline{C}(t_j - 0.5\Delta t_{eml}) \quad (5c)$$

$$Q_{l(k)} = \sum_{j=j_{start(k)}}^{j_{stop(k)}} y_{IF}(t_j)\,\overline{y}_Q(t_j)\,\overline{C}(t_j - 0.5\Delta t_{eml}) \quad (5d)$$

These are standard early and late in-phase and quadrature accumulations, which are also used in typical real-time receivers [2]. The interval Δt_{eml} is the delay, measured in seconds, between an early version of the reconstructed PRN code and a late version. The accumulation interval goes from $\overline{T}_{k-1}$ to $\overline{T}_k$, which implies that the sample index limits $j_{start(k)}$ and $j_{stop(k)}$ are chosen according to the rules

$$j_{start(k)} = \text{minimum } j \text{ such that } \overline{T}_{k-1} \leq t_j \quad (6a)$$

$$j_{stop(k)} = \text{maximum } j \text{ such that } t_j < \overline{T}_k \quad (6b)$$

Recall from eq. (2) that t_j, t_{j+1}, t_{j+2}, …etc… are the sample times of the bit grabber's ADC.

The accumulations in eqs. (5a)-(5d) can be used to compute carrier and code phase errors. The measured carrier phase error is

$$y_{carr(k)} =$$
$$\begin{cases} -\arctan2[Q_{e(k)}, I_{e(k)}] + n\pi & \text{if } [I_{l(k)}^2 + Q_{l(k)}^2] \leq [I_{e(k)}^2 + Q_{e(k)}^2] \\ -\arctan2[Q_{l(k)}, I_{l(k)}] + n\pi & \text{if } [I_{e(k)}^2 + Q_{e(k)}^2] < [I_{l(k)}^2 + Q_{l(k)}^2] \end{cases}$$

(7)

This quantity measures the difference between the true carrier phase perturbation, ϕ, and its reconstruction, ϕ_{re}. The integer n is selected to undo both the 2π phase ambiguity of the arctan2 function and the effects of GPS data bit shifts. This study assumes that the SNR at the sampling frequency is sufficiently high to allow for reliable determination of n by comparing $y_{carr(k)}$ with $y_{carr(k-1)}$; n gets adjusted to minimize the absolute value of the phase error change.

This phase error measurement is sub-optimal, and the method of computing n does not work well if the SNR is too low. Better techniques could be incorporated into this paper's developments, but such improvements would increase their complexity. The paper's goal is to introduce the concept of smoothing to the problem of GPS signal tracking. This goal is easier to accomplish if one keeps the smoothing algorithms as simple as possible. Equation (7) promotes simplicity because it gives rise to a linear carrier phase smoothing problem.

The measured code phase error is computed using a non-coherent calculation:

$$y_{code(k)} =$$
$$\left[\frac{2 - 1.023 \times 10^6 \Delta t_{eml}}{\beta\, 2.046 \times 10^6}\right] \frac{\sqrt{I_{e(k)}^2 + Q_{e(k)}^2} - \sqrt{I_{l(k)}^2 + Q_{l(k)}^2}}{\sqrt{I_{e(k)}^2 + Q_{e(k)}^2} + \sqrt{I_{l(k)}^2 + Q_{l(k)}^2}}$$

(8)

This phase error measures the difference between the estimated PRN code start time $\overline{T}$ and the actual code start time T. Equation (8) assumes a symmetric, triangular peak in the cross-correlation of the reconstructed code and the received code. The scalar β is nearly equal to 1 and accounts for slight variations in the slope of the autocorrelation function of different PRN codes.

Stochastic Carrier Phase Dynamics Model

A discrete-time carrier phase model has been developed which is similar to the one used in Ref. 14. It is a three-state discrete-time model:

$$\begin{bmatrix} x_p \\ x_v \\ x_a \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & \Delta T_k & \dfrac{\Delta T_k^2}{2} \\ 0 & 1 & \Delta T_k \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_p \\ x_v \\ x_a \end{bmatrix}_k - \begin{bmatrix} \Delta T_k \\ 0 \\ 0 \end{bmatrix} \omega_{re(k)}$$

$$+ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} w_k \qquad (9a)$$

$$y_{carr(k+1)} = \begin{bmatrix} 1 & \dfrac{\Delta T_k}{2} & \dfrac{\Delta T_k^2}{6} \end{bmatrix} \begin{bmatrix} x_p \\ x_v \\ x_a \end{bmatrix}_k - \dfrac{\Delta T_k}{2} \omega_{re(k)}$$

$$+ \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} w_k + v_{k+1} \qquad (9b)$$

The sample interval for this discrete-time model is $\Delta T_k = \overline{T}_{k+1} - \overline{T}_k$, which is nominally 0.001 sec. The model's three states are $x_p = \phi - \phi_{re}$, the carrier phase difference between the actual signal and the software receiver's reconstructed signal, $x_v = \dot{\phi}$, the carrier signal's Doppler shift, and $x_a = \ddot{\phi}$, the drift rate of the Doppler shift, which is caused by acceleration. The subscripts p, v, and a denote position, velocity, and acceleration.

The 4×1 vector w_k in eqs. (9a) and (9b) is the process disturbance vector. It models the effects of receiver vehicle maneuvers. It is a discrete-time Gaussian white noise process and has the following statistics:

$$E\{w_k\} = 0$$
$$E\{w_k w_l^T\}$$

$$= \delta_{kl} q_{ct} \begin{bmatrix} \Delta T_k^5/20 & \Delta T_k^4/8 & \Delta T_k^3/6 & \Delta T_k^5/72 \\ \Delta T_k^4/8 & \Delta T_k^3/3 & \Delta T_k^2/2 & \Delta T_k^4/30 \\ \Delta T_k^3/6 & \Delta T_k^2/2 & \Delta T_k & \Delta T_k^3/24 \\ \Delta T_k^5/72 & \Delta T_k^4/30 & \Delta T_k^3/24 & \Delta T_k^5/252 \end{bmatrix}$$

$$= \delta_{kl} Q_k \qquad (10)$$

In this equation δ_{kl} is the Kronecker delta, and q_{ct} is the intensity of an equivalent scalar continuous-time white noise process that models $\dddot{\phi}$. Equation (10) effectively defines the 4×4 discrete-time process noise intensity matrix Q_k.

Measurement eq. (9b) relates the states of the model in eq. (9a) to the carrier phase measurement that is defined in eq. (7). This relationship models $y_{carr(k+1)}$ as being the difference between $\phi(t)$ and $\phi_{re}(t)$ averaged over the accumulation interval $[\overline{T}_k, \overline{T}_{k+1})$.

Equation (9b) includes a measurement error term, v_{k+1}. This error is modeled as being a discrete-time white noise random process. It is assumed to be uncorrelated with the process noise vector, w_k, to be Gaussian, and to have the following statistics:

$$E\{v_{k+1}\} = 0, \qquad E\{v_{k+1}v_{l+1}^T\} = \delta_{kl}\,\sigma_v^2 \qquad (11)$$

This error term accounts for all measurement error sources other than vehicle dynamics. These include receiver thermal noise and digitization error, receiver clock jitter, etc. The form of this model is reasonable for errors with a wide spectrum, but it is less than optimal for low-frequency errors such as the ionospheric phase advance. In the future it may be possible to develop Kalman filters and smoothers that make use of error models which are more sophisticated.

The dynamic model in eqs. (9a) and (9b) can be put into standard matrix-vector form. Suppose that the model's 3×1 state vector is $x = [x_p;\ x_v;\ x_a]$. Then the model becomes:

$$x_{k+1} = \Phi_k x_k + \Gamma_k \omega_{re(k)} + \Gamma_w w_k \qquad (12a)$$

$$y_{carr(k+1)} = C_k x_k + D_k \omega_{re(k)} + D_w w_k + v_{k+1} \qquad (12b)$$

where the matrices Φ_k, Γ_k, Γ_w, C_k, D_k, and D_w are effectively defined in eqs. (9a) and (9b).

Stochastic Code Phase Dynamics Model

The code phase dynamics model is a first-order discrete-time model of the variations of the actual start times of the received signal's code periods:

$$T_{k+1} = T_k + \frac{0.001}{1 + (\dot{\phi}_k / \omega_{L1})} + w_{code(k)} \qquad (13a)$$

$$y_{code(k+1)} = \tfrac{1}{2}(\overline{T}_{k+1} + \overline{T}_k) - \tfrac{1}{2}(T_{k+1} + T_k) + v_{code(k+1)}$$
$$(13b)$$

In this model T_k, T_{k+1}, … etc. are the actual start times of the received PRN code periods. Recall that $\overline{T}_k$, $\overline{T}_{k+1}$, …, etc. are the start times of the receiver reconstruction of the code that is used to generate correlations. The second term on the right-hand side of eq. (13a) models the nominal PRN code period with an adjustment for Doppler effects. The quantity $\dot{\phi}_k$ is the average Doppler shift of the carrier signal during the time interval from T_k to T_{k+1}.

The scalars $w_{code(k)}$ and $v_{code(k+1)}$ are discrete-time white noise sequences, the former is called the process noise, and the latter is called the measurement noise. They are assumed to be Gaussian and uncorrelated with each other and to have the following statistics:

$$E\{w_{code(k)}\} = 0,$$
$$E\{w_{code(k)}w_{code(l)}^T\} = \delta_{kl}\,\sigma_{wcode}^2 \qquad (14a)$$

$$E\{v_{code(k+1)}\} = 0,$$
$$E\{v_{code(k+1)}v_{code(l+1)}^T\} = \delta_{kl}\,\sigma_{vcode}^2 \qquad (14b)$$

The process noise accounts for code/carrier divergence

that can be caused by ionospheric variations. The measurement noise accounts for receiver thermal noise, digitization error, and code multi-path error. The white-noise model for $v_{code(k+1)}$ is not totally consistent with code multi-path characteristics, but this is acceptable because Kalman filters and smoothers are often insensitive to such inconsistencies.

Equation (13b) relates the states T_k, T_{k+1}, … etc. of the dynamic model in eq. (13a) to the code phase measurement that is defined in eq. (8). Effectively, it says that the measured code phase at sample $k+1$ is the average over the accumulation interval $[\bar{T}_k, \bar{T}_{k+1})$ of the phase difference between the eq.-(3) reconstructed PRN code and the actual received PRN code.

KALMAN FILTERS AND IMPLEMENTATION OF PLL AND DLL FUNCTIONS

The main contribution of this work is in the area of GPS signal smoothing, but there are two good reasons also to consider the subject of Kalman filtering of GPS signals. First, Kalman filtering is closely related to smoothing. Second, Kalman filters have been used to design a PLL for tracking carrier phase and a DLL for tracking code phase. The PLL and the DLL are needed in order to get the receiver's replicas of the carrier and code to match closely with the received signal; otherwise, the linear models of this paper's third section would not be valid for purposes of smoothing.

Carrier Phase Kalman Filter and Associated PLL

The carrier phase Kalman filter produces the optimal estimate of the carrier phase state at time $\bar{T}_k$ based on the carrier phase measurements taken before and including that time. Suppose that this estimate is called $\tilde{x}_k$. Then the Kalman filter is based on eqs. (12a) and (12b), and it uses the following recursive formula to estimate $\tilde{x}_k$ starting from an initial estimate, $\tilde{x}_0$:

$$\tilde{v}_{k+1} = y_{carr(k+1)} - C_k \tilde{x}_k - D_k \omega_{re(k)} \tag{15a}$$

$$\tilde{x}_{k+1} = \Phi_k \tilde{x}_k + \Gamma_k \omega_{re(k)} + L_{k+1} \tilde{v}_{k+1} \tag{15b}$$

The quantity $\tilde{v}_{k+1}$ is called the filter innovation. It is the difference between the measured $y_{carr(k+1)}$ from eq. (7) and a prediction of $y_{carr(k+1)}$ that is based on the measured values of $y_{carr(0)}$, …, $y_{carr(k)}$. The time-varying L_{k+1} vector is the optimal Kalman filter gain [18].

A steady-state version of the Kalman filter in eqs. (15a) and (15b) can be used to develop a PLL. If ΔT_k is constant, which is a good approximation, then the Kalman filter gain approaches a constant as k gets large; i.e., $L_{k+1} \to L$ as $k \to \infty$. This constant gain is used in the PLL. The PLL feeds back the estimated carrier phase state at

time $\bar{T}_{k+1}$ to select the carrier reconstruction frequency for the time interval $\bar{T}_{k+2}$ to $\bar{T}_{k+3}$:

$$\omega_{re(k+2)} = \{ [(1-\alpha)^2, (\{1-2\alpha\}\Delta T_{k+1} + \Delta T_{k+2}),$$
$$0.5(-2\alpha\Delta T_{k+1}^2 + \{\Delta T_{k+1} + \Delta T_{k+2}\}^2)] \tilde{x}_{k+1}$$
$$- (1-\alpha)^2 x_{peq} - (1-2\alpha) \omega_{re(k+1)} \Delta T_{k+1} \}/\Delta T_{k+2} \tag{16}$$

The scalar x_{peq} is a desired equilibrium value of x_p, the phase difference between the reconstructed carrier signal and the actual carrier signal. The quantity α is a tuning parameter that is in the range $0 \le \alpha < 1$. Assuming that the Kalman filter's phase estimate is correct, α determines how fast x_p will converge to x_{peq}: $\alpha = 0$ causes convergence in two code periods, but α near 1 yields very slow convergence. Although not needed in the current application, the use of $\tilde{x}_{k+1}$ to determine $\omega_{re(k+2)}$ allows for causal operation of the PLL even when one considers its computational delay.

The x_{peq} bias term is set to $\pm\pi/2$, whichever is nearer to the initial phase error. This bias has been added in order to keep the PLL's steady-state response as far away as possible from the $\pm\pi$ ambiguity of the arctan2 function in eq. (7). If $x_{peq} = 0$ were used, which is typical, then data bit shifts would place the eq. (7) calculation near to the ambiguity half of the time.

The entire PLL consists of eqs. (15a), (15b), and (16). It is stable for reasonable choices of L. It is 5th-order, but α is normally chosen to be small enough to cause the PLL's response to be dominated by the Kalman filter. This makes the PLL effectively 3rd order.

This PLL is used by the software receiver to determine $\omega_{re(2)}$, $\omega_{re(3)}$, $\omega_{re(4)}$, … etc. These quantities are needed in order to set up a linear smoothing problem. The values of $\omega_{re(0)}$ and $\omega_{re(1)}$ are needed to initialize the PLL. They are determined by the signal acquisition process.

Code Phase Kalman Filter and Associated DLL

A Kalman filter can also be developed to estimate the code phase. It is based on the code phase model in eqs. (13a) and (13b). It takes the following recursive form

$$\tilde{v}_{code(k+1)} = y_{code(k+1)} - \frac{1}{2}(\bar{T}_{k+1} + \bar{T}_k) + \tilde{T}_k$$
$$+ \frac{1}{2}\left\{ \frac{0.001}{1 + ([0,1,0.0005]\tilde{x}_k / \omega_{L1})} \right\} \tag{17a}$$

$$\tilde{T}_{k+1} = \tilde{T}_k + \frac{0.001}{1 + ([0,1,0.0005]\tilde{x}_k / \omega_{L1})}$$
$$+ L_{code(k+1)} \tilde{v}_{code(k+1)} \tag{17b}$$

The scalar $\tilde{T}_k$ is the optimal estimate of T_k based on the

measurements $y_{code(0)}$, $y_{code(1)}$, $y_{code(2)}$, …, $y_{code(k)}$. The quantity $\tilde{v}_{code(k+1)}$ is the code phase innovation, and $L_{code(k+1)}$ is the code phase Kalman filter gain.

This Kalman filter uses carrier aiding. The aiding term uses an estimate of the average carrier Doppler shift for the time interval $[\bar{T}_k, \bar{T}_{k+1})$. This estimate is $[0, 1, 0.0005]\tilde{x}_k$.

A steady-state version of the code phase Kalman filter can be used as the basis for a DLL. The steady-state filter uses a constant L_{code} filter gain that is the asymptotic limit of $L_{code(k+1)}$ as k approaches infinity. The DLL is completed by inclusion of a "feedback control law" that computes future values of $\bar{T}$ based on current estimates $\tilde{T}$. The feedback control law computes

$$\bar{T}_{k+3} = \tilde{T}_{k+1} + \frac{0.001}{1+([0,1,0.0005]\tilde{x}_{k+1}/\omega_{L1})}$$
$$+ \frac{0.001}{1+([0,1,0.0015]\tilde{x}_{k+1}/\omega_{L1})} \qquad (18)$$

The last two terms on the right-hand side of this equation use aiding from the carrier phase Kalman filter. This formula for $\bar{T}_{k+3}$ defines how the DLL regulates the PRN chipping rate for the interval $[\bar{T}_{k+2}, \bar{T}_{k+3})$; it sets it to $1023/(\bar{T}_{k+3}-\bar{T}_{k+2})$. Equations (17a), (17b), (18) and the chipping rate formula constitute the complete DLL. The control law assumes that the DLL calculations take place in real time and during the time interval from $\bar{T}_{k+1}$ to $\bar{T}_{k+2}$. That is why it uses $\tilde{T}_{k+1}$ to determine $\bar{T}_{k+3}$ rather than $\bar{T}_{k+2}$. After $\bar{T}_{k+1}$ has passed, the receiver assumes that the code chipping rate remains fixed until time $\bar{T}_{k+2}$, which means that eq. (18) executes too late to affect $\bar{T}_{k+2}$.

The software receiver uses this DLL. It needs accurate values for $\bar{T}_0$, $\bar{T}_1$, $\bar{T}_2$, … in order to set up its linear smoothing problem. The signal acquisition algorithm can be used to estimate $\bar{T}_0$ and $\bar{T}_1$ to sufficient accuracy, but the DLL is needed in order to estimate $\bar{T}_2$, $\bar{T}_3$, $\bar{T}_4$, …

SMOOTHERS THAT TRACK CARRIER PHASE AND CODE PHASE

A fixed-interval smoother processes a batch of data that extends over a given time interval and optimally estimates the state over that entire time interval based on the entire data batch. The resulting estimated state time history is more accurate than that of a Kalman filter. The accuracy increases because the estimate at any given time is based on a larger data set.

Carrier Phase Smoother

The following is a least squares formulation of the carrier phase smoothing problem:

find: x_k for $k = 0, …, N$
            w_k and v_{k+1} for $k = 0, …, N$-1   (19a)

to minimize:
$$J = \frac{1}{2}(\tilde{R}_0 x_0 - \tilde{z}_0)^T(\tilde{R}_0 x_0 - \tilde{z}_0)$$
$$+ \frac{1}{2}\sum_{k=0}^{N-1}(R_{w(k)}w_k)^T(R_{w(k)}w_k)$$
$$+ \frac{1}{2}\sum_{k=1}^{N}(R_v v_k)^T(R_v v_k) \qquad (19b)$$

subject to: $x_{k+1} = \Phi_k x_k + \Gamma_k \omega_{re(k)} + \Gamma_w w_k$
             for $k = 0, …, N$-1   (19c)

    $y_{carr(k+1)} = C_k x_k + D_k \omega_{re(k)} + D_w w_k + v_{k+1}$
             for $k = 0, …, N$-1   (19d)

were eqs. (19c) and (19d) repeat the carrier phase dynamic model of eqs. (12a) and (12b). This problem seeks an optimal state time history, x_0, …, x_N, an optimal process noise time history, w_0, …, w_{N-1}, and an optimal measurement error time history, v_1, …, v_N. These optimal time histories must satisfy the dynamic model in eq. (19c) and must reproduce the measurements $y_{carr(1)}$, …, $y_{carr(N)}$ in accordance with the measurement model in eq. (19d). They also must minimize the weighted sum of the squares of the process-noise and measurement-error vectors along with a weighted sum of the difference between x_0 and its *a priori* estimate $\tilde{x}_0 = \tilde{R}_0^{-1}\tilde{z}_0$.

The given quantities of this problem are the scalar R_v, the scalars D_k and $y_{carr(k+1)}$ for $k = 0, …, N$-1, the vector $\tilde{z}_0$, the matrices $\tilde{R}_0$, Γ_w, and D_w, and the matrices $R_{w(k)}$, Φ_k, Γ_k, and C_k for $k = 0, …, N$-1. The only quantities in this set that have not been defined already are those which weight the errors in the eq.-(19b) least-squares cost function. Each of these quantities has a statistical definition. The 3×3 matrix $\tilde{R}_0$ is the square root of the inverse of the estimation error covariance of $\tilde{x}_0$: $\tilde{R}_0^{-1}\tilde{R}_0^{-T} = E\{(\tilde{x}_0 - x_0)(\tilde{x}_0 - x_0)^T\}$, where the notation $()^{-T}$ stands for the transpose of the inverse of the matrix in question. The 3×1 vector $\tilde{z}_0 = \tilde{R}_0 \tilde{x}_0$. The 4×4 matrix $R_{w(k)}$ is the square root of the inverse of the process noise covariance matrix: $R_{w(k)}^{-1}R_{w(k)}^{-T} = Q_k$, where Q_k comes from eq. (10). The scalar $R_v = 1/\sigma_v$, the inverse of the standard deviation of the measurement error.

This smoothing problem can be solved by using a modified form of the standard square-root information filter/covariance smoother (SRIF/S) algorithm [12]. The modified algorithm begins with a manipulation of

measurement eq. (19d). It subtracts $D_k\,\omega_{re(k)}$ from both sides and then multiplies both sides by R_V. The result is

$$z_{m(k)} = A_k x_k + A_w w_k + V_{m(k)} \qquad \text{for } k = 0, \ldots, N\text{-1} \quad (20)$$

where $z_{m(k)} = R_V\,[y_{carr(k+1)} - D_k\omega_{re(k)}]$, $A_k = R_V\,C_k$, $A_w = R_V\,D_w$, and $V_{m(k)} = R_V\,V_{k+1}$.

The following steps define the smoothing algorithm:

1. Set $k = 0$.

2. Use left QR factorization to compute Q_{1k}, $\bar{R}_{ww(k)}$, $\bar{R}_{wx(k)}$, and $\hat{R}_k$ such that $Q_{1k}Q_{1k}^T = I$ and

$$Q_{1k}^T \begin{bmatrix} \bar{R}_{ww(k)} & \bar{R}_{wx(k)} \\ 0 & \hat{R}_k \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} R_w & 0 \\ A_w & A_k \\ 0 & \tilde{R}_k \end{bmatrix}$$

3. Compute $\begin{bmatrix} \bar{z}_{w(k)} \\ \hat{z}_k \\ \bar{z}_{e(k)} \end{bmatrix} = Q_{1k} \begin{bmatrix} 0 \\ z_{m(k)} \\ \tilde{z}_k \end{bmatrix}$

4. If $k = N\text{-1}$ go to Step 8.

5. Use left QR factorization to compute Q_{2k}, $\hat{R}_{ww(k)}$, $\hat{R}_{wx(k)}$, and $\tilde{R}_{k+1}$ such that $Q_{2k}Q_{2k}^T = I$ and

$$Q_{2k}^T \begin{bmatrix} \hat{R}_{ww(k)} & \hat{R}_{wx(k)} \\ 0 & \tilde{R}_{k+1} \end{bmatrix} =$$
$$\begin{bmatrix} (\bar{R}_{ww(k)} - \bar{R}_{wx(k)}\Phi_k^{-1}\Gamma_w) & \bar{R}_{wx(k)}\Phi_k^{-1} \\ -\hat{R}_k\Phi_k^{-1}\Gamma_w & \hat{R}_k\Phi_k^{-1} \end{bmatrix}$$

6. Compute $\begin{bmatrix} \hat{z}_{w(k)} \\ \tilde{z}_{k+1} \end{bmatrix} =$
$$Q_{2k}\begin{bmatrix} (\bar{z}_{w(k)} + \bar{R}_{wx(k)}\Phi_k^{-1}\Gamma_k\omega_{re(k)}) \\ (\hat{z}_k + \hat{R}_k\Phi_k^{-1}\Gamma_k\omega_{re(k)}) \end{bmatrix}$$

7. Replace k with $k+1$ and go to Step 2.

8. Compute $x_{N-1}^* = \hat{R}_{N-1}^{-1}\hat{z}_{N-1}$,
   $w_{N-1}^* = \bar{R}_{ww(N-1)}^{-1}[\bar{z}_{w(N-1)} - \bar{R}_{wx(N-1)}x_{N-1}^*]$,
   and $x_N^* = \Phi_{N-1}x_{N-1}^* + \Gamma_{N-1}\omega_{re(N-1)} + \Gamma_w w_{N-1}^*$ and set $k = N\text{-2}$.

9. Compute $w_k^* = \hat{R}_{ww(k)}^{-1}[\hat{z}_{w(k)} - \hat{R}_{wx(k)}x_{k+1}^*]$ and
   $x_k^* = \Phi_k^{-1}[x_{k+1}^* - \Gamma_k\omega_{re(k)} - \Gamma_w w_k^*]$

10. If $k = 0$ stop; otherwise, replace k with $k\text{-1}$ and go to Step 9.

In this algorithm $x_0^*, \ldots, x_N^*$ is the smoothed state time history, and $w_0^*, \ldots, w_{N-1}^*$ is the smoothed process noise time history. The smoothed measurement error time history, $v_1^*, \ldots, v_N^*$, can be computed by using eq. (19d).

The smoother consists of a forward pass through the data, Steps 1-7, followed by a backwards recursion, Steps 8-10. The forward pass is equivalent to the Kalman filter of eqs. (15a) and (15b) [12].

This smoother can be tuned by selecting the various statistical weighting matrices in the eq.-(19b) least-squares cost function. The $\tilde{R}_0$ matrix affects the initial transient behavior of the smoother near the start of the data batch. A large value of $\tilde{R}_0$ causes the smoother to rely more on the *a priori* guess $\tilde{x}_0$ than on the measured data for small values of k. In the current application $\tilde{R}_0$ is set to zero. This causes the smoother to ignore the initial guess of the state and to determine the state based only on the measurements. The other tuning parameters are the process noise intensity, q_{ct} from eq. (10), and the measurement noise standard deviation, σ_v from eq. (11). A high value of q_{ct} or a low value of σ_v will cause the smoother to form its x_k^* estimate mostly based on data taken very near time $\bar{T}_k$. Conversely, a low value of q_{ct} or a high value of σ_v will cause x_k^* to be based on a long window of data, which will get mapped to time $\bar{T}_k$ by using the dynamic model in eq. (19c).

Code Phase Smoother

The code phase smoothing problem has a least-squares formulation which is similar to that of the carrier phase problem:

find: T_k for $k = 0, \ldots, N$
$w_{code(k)}$ and $v_{code(k+1)}$ for $k = 0, \ldots, N\text{-1}$
$$(21a)$$

to minimize: $J = \frac{1}{2}(\tilde{r}_0 T_0 - \tilde{z}_{T(0)})^2$
$$+ \frac{1}{2}\sum_{k=0}^{N-1}(w_{code(k)}/\sigma_{wcode})^2$$
$$+ \frac{1}{2}\sum_{k=1}^{N}(v_{code(k)}/\sigma_{vcode})^2 \qquad (21b)$$

subject to: $T_{k+1} = T_k$
$$+ \frac{0.001}{1 + \{[0, 0.5, 0](x_k^* + x_{k+1}^*)/\omega_{L1}\}}$$
$$+ w_{code(k)} \qquad \text{for } k = 0,\ldots,N\text{-1} \quad (21c)$$

$$y_{code(k+1)} = \frac{1}{2}(\bar{T}_{k+1} + \bar{T}_k) - \frac{1}{2}(T_{k+1} + T_k)$$
$$+ v_{code(k+1)} \qquad \text{for } k = 0,\ldots,N\text{-1} \quad (21d)$$

This smoothing problem has a scalar state, T_k. The quantities $\tilde{r}_0$ and $\tilde{z}_{T(0)}$ are defined in terms of the statistics of $\tilde{T}_0$, which is the *a priori* estimate of the start time of the initial code period. $1/\tilde{r}_0$ is the *a priori* standard deviation of the estimation error in $\tilde{T}_0$, and $\tilde{z}_{T(0)} = \tilde{r}_0 \tilde{T}_0$.

This smoothing problem incorporates aiding from the carrier phase smoother. Carrier aiding affects the second term on the right-hand side of eq. (21c).

Like the carrier phase smoother, the code phase smoother is a modified version of the standard SRIF/S algorithm of Ref. 12. It takes the form:

1. Set $k = 0$.

2. Use left QR factorization to compute $Q_{1co(k)}$, $\hat{r}_k$, $\hat{s}_{k+1}$, and $\tilde{r}_{k+1}$ such that $Q_{1co(k)}Q_{1co(k)}^{\mathrm{T}} = I$ and

$$Q_{1co(k)}^{\mathrm{T}} \begin{bmatrix} \hat{r}_k & \hat{s}_{k+1} \\ 0 & \tilde{r}_{k+1} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \tilde{r}_k & 0 \\ 1/\sigma_{wcode} & -1/\sigma_{wcode} \\ 1/(2\sigma_{vcode}) & 1/(2\sigma_{vcode}) \end{bmatrix}$$

3. Compute $\begin{bmatrix} \hat{z}_{T(k)} \\ \tilde{z}_{T(k+1)} \\ \bar{z}_{eco(k)} \end{bmatrix} =$

$$Q_{1co(k)} \begin{bmatrix} \tilde{z}_{T(k)} \\ \dfrac{1}{\sigma_{wcode}} \left\{ \dfrac{-0.001}{1 + \{[0,0.5,0](x_k^* + x_{k+1}^*)/\omega_{L1}\}} \right\} \\ \dfrac{1}{\sigma_{vcode}} \left\{ \tfrac{1}{2}(\bar{T}_{k+1} + \bar{T}_k) - y_{code(k+1)} \right\} \end{bmatrix}$$

4. If $k = N$-1 go to Step 6.

5. Replace k with $k+1$ and go to Step 2.

6. Compute $T_N^* = \tilde{z}_{T(N)} / \tilde{r}_N$.

7. Set $k = N$-1.

8. Compute $T_k^* = [\hat{z}_{T(k)} - \hat{s}_{k+1} T_{k+1}^*]/\hat{r}_k$.

9. If $k = 0$ stop; otherwise, replace k with k-1 and go to Step 8.

The quantities T_0^*, ..., T_N^* are the smoothed estimates of the PRN code period start times. The smoothed estimates $w_{code(k)}^*$ and $v_{code(k+1)}^*$ can be determined from eqs. (21c) and (21d).

This smoother is similar to the carrier phase smoother. It implements a forward iteration, Steps 1-5, followed by a backwards iteration, Steps 6-9. The forward iteration is equivalent to the code phase Kalman filter of eqs. (17a) and (17b). Tuning is accomplished by adjusting $\tilde{r}_0$, σ_{wcode}, and σ_{vcode}. A value of $\tilde{r}_0 = 0$ has been assumed so that T_0^* will be based only on measurement data with no influence from an *a priori* guess of $\tilde{T}_0$.

Checking the Smoothed Results via Re-Correlation

The smoothers' performance can be checked by recalculation of the following quantities: the correlations in eqs. (5a)-(5d), the measured carrier phase error in eq. (7), and the measured code phase error in eq. (8). If the smoother is functioning properly, then the re-calculated phase error measurements will not display transient behavior. The re-correlation process uses the smoother outputs to update its replicas of the received carrier and PRN code signals. The PRN code signal replica gets updated by replacing $\bar{T}_k$ with T_k^* for $k = 0, ..., N$. The carrier signal gets updated by creating a new $\phi_{re(k)}$ that is equal to the original value plus the correction term $\{[1, 0, 0] x_k^* - x_{peq}\}$. In addition, the new $\phi_{re(k)}$ gets interpolated in order to transfer its definition from the DLL-generated time grid point $\bar{T}_k$ to the smoothed time grid point T_k^*. As part of this process, $\omega_{re(k)}$ gets changed in order to maintain its defined relationship to $\phi_{re(k)}$.

Post-Processing of Data from a Conventional Real-Time Receiver

These smoothing algorithms could be used to post-process data from a real-time receiver, which would improve signal tracking accuracy. The real-time receiver would have to store the following quantities for later processing: $\bar{T}_0$, $\bar{T}_1$, $\bar{T}_2$,..., $\bar{T}_N$, $\phi_{re(0)}$, $\phi_{re(1)}$, $\phi_{re(2)}$, ..., $\phi_{re(N)}$, $y_{carr(1)}$, $y_{carr(2)}$, $y_{carr(3)}$, ..., $y_{carr(N)}$, and $y_{code(1)}$, $y_{code(2)}$, $y_{code(3)}$, ..., $y_{code(N)}$. Alternatively, accumulations like those of eqs. (5a)-(5d) could be stored in lieu of $y_{code(k)}$ and $y_{carr(k)}$. Post processing will be effective only if the receiver has generated the $\bar{T}_k$ and $\phi_{re(k)}$ time histories in such a way that it maintains lock on the signal.

EXPERIMENTAL RESULTS

The Kalman filters and smoothers of this paper's fourth and fifth sections have been applied to track actual GPS signals. These signals have been recorded using the bit grabber that is described in this paper's second section. The nominal carrier phase Kalman filter/smoother tuning parameters are $q_{ct} = 1,300$ rad^2/sec^5 and $\sigma_v = 0.114$ rad. The nominal tuning parameters for the code-phase Kalman filter/smoother are $\sigma_{wcode} = 2.55 \times 10^{-10}$ sec and $\sigma_{vcode} = 4.06 \times 10^{-8}$ sec. These tuning parameters lead to a carrier-tracking PLL bandwidth of 3.42 Hz with $L =$

[0.043; 0.913; 9.787]. The nominal code-tracking DLL bandwidth is 1 Hz with $L_{code} = -0.00626$.

Carrier Phase Tracking Results

The Doppler shift estimates for a typical case are shown in Fig. 2. The Kalman filter starts with an initial Doppler shift estimate of 1700 Hz, but the actual Doppler shift is 1780 Hz. The Kalman filter-based PLL takes 0.4 sec to converge to the true Doppler shift. The smoother, on the other hand, has no convergence transient. It correctly estimates the Doppler shift for the entire interval as being 1780 Hz. This lack of a convergence transient implies that there is no inherent phase lag between a smoother's Doppler shift estimate and the actual Doppler shift.

The smoother also produces a less noisy Doppler shift estimate. Figure 3 presents another comparison between a Kalman-filter-based Doppler shift estimate and a smoother-based estimate. The Kalman filter for this case starts with very accurate *a priori* values for the carrier signal's initial phase and Doppler shift. This eliminates transient error effects. Even so, the Kalman-filter-generated estimate obviously contains significantly more high-frequency noise than does the smoother's estimate. Most of the Kalman filter's dynamic variations on this plot are caused by receiver noise, while most of the smoother's variations constitute a real signal, perhaps the Selective Availability (SA) signal. This data was collected during the second half of 1999, before SA was turned off.

It is possible to analytically compare the Kalman filter and the smoother in terms of their effective SNRs. If one neglects the effects of initial transients in the Kalman filter and of end conditions in the smoother, then each of these estimators can be recast in the following form:

$$x_k = \sum_{l=1}^{N} \alpha_{k-l} y_{carr(l)} + \sum_{l=0}^{N-1} \gamma_{k-l} \omega_{re(l)}$$
$$\text{for } k = 0,\ldots,N \qquad (22)$$

where the 3×1 vectors α_{k-l} and γ_{k-l} are influence coefficient distributions. Figure 4 plots the second element of α_{k-l} vs. the time offset $-(k-l)\Delta T$. As can be seen in the figure, the Kalman filter's distribution is one-sided, on the lagging side of the estimation point. This reflects its causal nature. The smoother's distribution is asymmetric about the estimation point and has no time lag. The asymmetry is caused by the need to differentiate the carrier phase measurement in order to determine the Doppler shift. Both distributions have about the same time width, which means that they are roughly equivalent in terms of the signal bandwidth that they can successfully track.
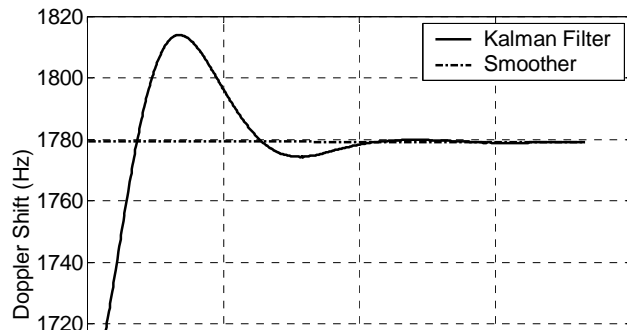
Fig. 2. Estimated Doppler shift time histories from the Kalman filter and from the smoother, PRN 14.

The SNR performance improvement of a smoother can be understood in terms of these influence coefficients. The SNR of each estimator varies inversely with $\sum \alpha_{k-l}^2$, the sum of the squares of the influence coefficients. Therefore, according to the data in Fig. 4, the smoother's Doppler estimate has an SNR that is 12.5 dB larger than that of the Kalman filter. This explains why the smoother's Doppler shift curve in Fig. 3 is so much less noisy than that of the Kalman filter. Although not shown, the smoother exhibits similar SNR performance improvements in its x_p carrier phase estimates (7.8 dB of improvement over the Kalman filter) and in its x_a Doppler drift rate estimates (7.7 dB of improvement). These performance improvements will change if the tuning parameters q_{ct} and σ_v get changed.

Figure 5 further illustrates the performance of the carrier phase smoother by plotting the measured phase difference between the received carrier signal and its smoothed replica. The vertical axis is essentially y_{carr} from eq. (7), but with n set to 0. This plot clearly shows the effects of navigation data bit transitions. These are the 0.5-cycle jumps that occur at regular multiples of 20 msec. This plot illustrates the smoother's lack of an initial transient, and it shows the effects of thermal noise. Noise causes the significant phase fluctuations that occur between the bit transitions. This type of noise is what caused the high-frequency errors in the Kalman filter's Doppler shift estimate on Fig. 3. The smoother also has to contend with this noise, but it does a much better job of attenuating the noise through signal processing.
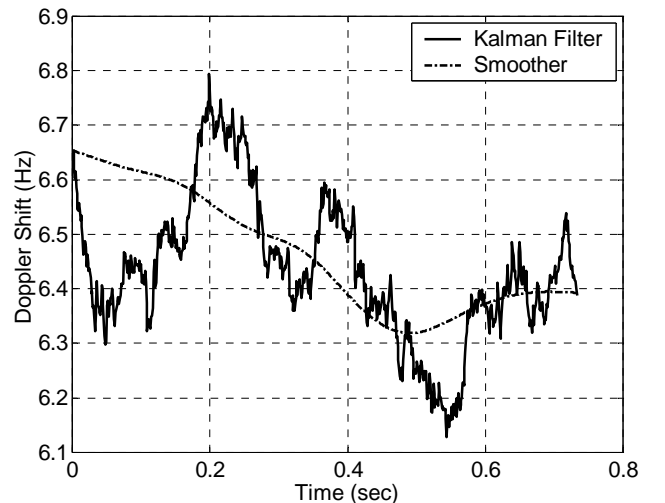
Code Phase Tracking Results

The operation of the code phase Kalman filter and its associated smoother are illustrated by the results of Fig. 6. The solid curve on this figure is the time history of the Kalman-filter-based DLL's estimated C/A code period minus the nominal code period, ($\bar{T}_{k+1} - \bar{T}_k$ - 0.001). The figure's dashed-dotted curve is the corresponding quantity for the smoother, ($T^*_{k+1} - T^*_k$ - 0.001). The Kalman filter's estimated code period experiences an initial transient decay that is caused by a code phase error from the acquisition module. The smoother, on the other hand, has no such transient. Also, the smoother's code period is much less noisy.
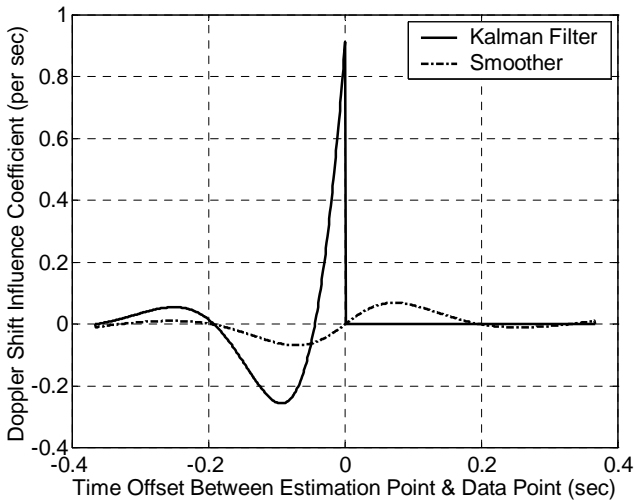

Fig. 4.   Time histories of effective influence coefficients for Kalman filter and smoother estimation of the Doppler shift.
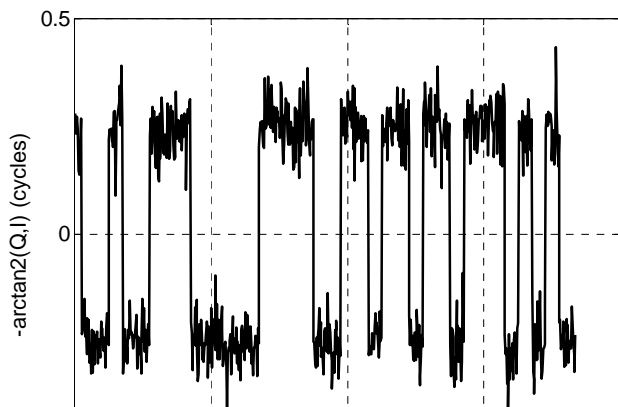

Fig. 5.   Measured carrier phase difference between received signal and smoothed replica signal, PRN 16.

The relative SNRs of the code phase Kalman filter and the code phase smoother have been analyzed. The analysis has computed influence coefficients much like the carrier phase influence coefficients of eq. (22) and Fig. 4. This analysis shows that the smoother estimates the code phase with an SNR that is 3 dB larger than that of the Kalman filter. The SNR of the smoother's code period estimate is 28 dB larger than that of the Kalman filter, which explains the difference in the noisiness of the two curves on Fig. 6. As in the case of carrier phase, the code phase smoother achieves its SNR improvements without a significant loss of signal tracking bandwidth in comparison to the Kalman filter. Similarly, the code phase smoother does not have the Kalman filter's phase
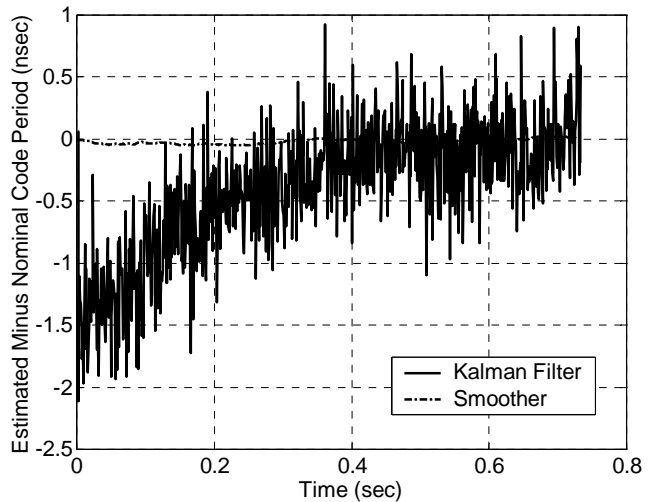

Fig. 6.   Time histories of C/A code period offsets from 0.001 sec for a code phase Kalman filter and for a code phase smoother, PRN 25.

Computational Cost

The computational costs of these algorithms are reasonable for a post-processing environment. They scale linearly with N, the number of code periods. This scaling is the result of the Kalman filters' and smoothers' efficient recursive implementations.

The actual time to run these algorithms has been recorded

for a particular computing platform. It is a 400 MHz Pentium machine that runs the Windows NT operating system. The algorithms have been encoded and executed in MATLAB. The following execution speeds have been achieved: The Kalman filtering with correlations required 56.6 sec of computation time per second of bit-grabbed receiver data. The smoothing with re-correlation used 58.9 sec of computation per second of bit-grabbed data. The majority of this time was used to compute the correlations in eqs. (5a)-(5d).

This computation time could be improved by using a language other than MATLAB or by using compiled MATLAB. MATLAB has been run in its interpretive form during this study. Equivalent compiled code might execute as much as 10 times faster.

OPEN QUESTIONS ABOUT SMOOTHER-BASED GPS SIGNAL TRACKING

This is study represents a first cut at the use of smoothing to track a GPS signal in a software receiver. Many possible smoother refinements have not been studied in the interests of maintaining a reasonable scope for this paper. In the future, however, it would be good to consider various possible enhancements to smoother-based GPS signal tracking.

One natural enhancement would be to let the smoother do optimal data demodulation along with optimal determination of the modulo-20 code period at which data bit shifts can occur. The present paper assumed that the received SNR was high enough to allow the use of ad hoc methods to detect the data bit shifts. One of the important applications of a smoother is to the case of a very low received SNR, which causes the issue of optimal data bit estimation to become important. One possible approach to this problem is to use multiple-model estimation techniques in conjunction with integer least-squares techniques. A research challenge will be to adapt integer programming techniques to the iterative framework of a Kalman-filter/smoother.

Another important set of issues involves cycle slips and loss of signal lock. The current algorithms assume that neither problem occurs. These problems tend to arise in applications with low SNR or high dynamics. Cycle slips and loss of lock can be addressed in a Kalman-filter/smoother framework, but significant enhancements to the current algorithms would be needed. One way to deal with these issues is to design a nonlinear filter/smoother that employs an iterative nonlinear least-squares technique such as the Gauss-Newton method. Such a nonlinear estimator might employ the present smoother as a means of determining a search direction in phase-time-history space. It would iteratively search for the optimum of a nonlinear estimation performance index.

Another interesting issue is that of the code/carrier divergence, which can be caused by the ionosphere. The smoother could be adapted to look for this. The adaptation might involve integration of the carrier-phase and code-phase algorithms into a single coupled smoother. In this case, the measurement error models would be modified to include the negative correlation that exists between the ionosphere's effect on code phase and its effect on carrier phase.

Smoother-based signal tracking also could be developed for a dual-frequency receiver. If the receiver were a military receiver with access to the encrypted P(Y) code, then the present developments should adapt in a straightforward manner. If, however, the receiver did not have access to the P(Y) code, then the smoother would have to be based on cross-correlation between the *L1* and *L2* versions of the code. The development of such a cross-correlation-based smoother would require some further thought. Of course, any dual-frequency work would require a dual frequency bit grabber that could sample at about 5 times the P(Y) code chipping rate, i.e., at about 50 MHz.

SUMMARY AND CONCLUSIONS

New signal tracking algorithms have been designed for use in a non-real-time software GPS receiver. These algorithms perform PRN code phase tracking and carrier phase tracking for the C/A civilian signal. The code phase tracking is performed in two steps. The first step implements a Kalman-filter-based delay-locked loop. It tracks the code phase to within the linear region of its discriminator function. The second step uses after-the-fact data in a non-causal square-root information filter/smoother in order to refine the code phase estimates. A random-walk process and carrier aiding have been included in the dynamic code phase model that gets used by the filter and the smoother. The carrier phase tracking loop works similarly. It uses a Kalman-filter-based phase-locked loop to perform rough-cut signal tracking and a non-causal smoother to make the final carrier phase estimate. These latter two algorithms use a carrier phase signal model that is a cascade of 3 integrators driven by white noise.

The two smoothers make optimal use of correlation-based phase measurements. They are non-causal because they use data from before and after a given time of interest. This allows them to eliminate the filtering lags that are normally associated with phase-locked loops and delay-locked loops. Also, they significantly increase the SNR without decreasing their tracking bandwidth. The only drawback of these smoothers is that they cannot be used in real time. Fortunately, there are significant applications that do not require real-time operation.

These new signal tracking algorithms have been tested using experimental C/A GPS data. The data has been collected using a bit-grabber/digital storage receiver that was connected to a roof-top antenna. Test results show that the smoothers can track code phase and carrier phase with good accuracy. A carrier phase smoother with a 3 Hz bandwidth has a phase estimation SNR that is 7.8 dB higher than for an equivalent Kalman-filter-based PLL, and its Doppler shift SNR is 12.5 dB higher. A code phase smoother with a 1 Hz bandwidth has a code phase SNR that is 3 dB higher than that of the corresponding Kalman-filter-based delay-locked loop, and its code period SNR is 28 dB higher. The bottom line is that smoothers offer significant SNR improvements and the ability to track dynamic signals without introducing an estimation lag.

ACKNOWLEDGMENT

References

1. Spilker, J.J. Jr., "Fundamentals of Signal Tracking Theory," in *Global Positioning System: Theory and Applications, Vol. I*, Parkinson, B.W. and Spilker, J.J. Jr., eds., American Institute of Aeronautics and Astronautics, (Washington, 1996), pp. 245-327.

2. Van Dierendonck, A.J., "GPS Receivers," in *Global Positioning System: Theory and Applications, Vol. I*, Parkinson, B.W. and Spilker, J.J. Jr., eds., American Institute of Aeronautics and Astronautics, (Washington, 1996), pp. 329-407.

3. Akos, D.M., and Tsui, J.B.Y., "Design and Implementation of a Direct Digitization GPS Receiver Front End," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 44, No. 12, Dec., 1996, pp. 2334-2339.

4. Reichert, A., Axelrad, P., Wu, S.C., Bertiger, W., and Srinivasan, J., "Initial Demonstration of a Point Solution Algorithm for Orbit Determination Using the microGPS Receiver," *Proceedings of the ION National Technical Meeting*, Institute of Navigation, Alexandria, Virginia, Jan., 1997, pp. 377-386.

5. Tsui, J.B.Y., Stockmaster, M.H., and Akos, D.M., "Block Adjustment of Synchronizing Signal (BASS) for Global Positioning System (GPS) Receiver Signal Processing," *Proceedings of the ION GPS '97*, Institute of Navigation, Alexandria, Virginia, Sept., 1997, pp. 637-641.

6. Srinivasan, J., Bar-Sever, Y., Bertiger, W., Lichten, S., Muellerschoen, R., Munson, T., Spitzmesser, D., Tien, J., Wu, S.C., and Young, L., "microGPS: On-Orbit Demonstration of a New Approach to GPS for Space Applications," *Proceedings of the ION GPS '98*, Institute of Navigation, Alexandria, Virginia, Sept., 1998, pp. 1537-1545.

7. Snyder, C.A., Feng, G., and van Graas, F., "GPS Anomalous Event Monitor (GAEM)," *Proceedings of the ION 55th Annual Meeting*, Institute of Navigation, Alexandria, Virginia, June, 1999, pp. 185-189.

8. Feng, G., and van Graas, F., "GPS Receiver Block Processing," *Proceedings of the ION GPS '99*, Institute of Navigation, Alexandria, Virginia, Sept., 1999, pp. 307-315.

9. Schamus, J.J., and Tsui, J.B.Y., "Acquisition to Tracking and Coasting for Software GPS Receiver," *Proceedings of the ION GPS '99*, Institute of Navigation, Alexandria, Virginia, Sept., 1999, pp. 325-328.

10. May, M., Brown, A., and Tanju, B., "Applications of Digital Storage Receivers for Enhanced Signal Processing," *Proceedings of the ION GPS '99*, Institute of Navigation, Alexandria, Virginia, Sept., 1999, pp. 2199-2208.

11. Fridman, A., and Semenov, S., "Architectures of Software GPS Receivers," *GPS Solutions*, Vol. 3, No. 4, Spring, 2000, pp. 58-64.

12. Bierman, G.J., *Factorization Methods for Discrete Sequential Estimation*, Academic Press, (New York, 1977), pp. 57-67, 69-76, 115-122, and 214-217.

13. Gustafson, D.E., "GPS Signal Tracking Using Maximum-Likelihood Parameter Estimation," *Navigation: Journal of the Institute of Navigation*, Vol. 45. No. 4, Winter, 1998-1999, pp. 287-295.

14. Psiaki, M.L., "Attitude Sensing Using a Global-Positioning-System Antenna on a Turntable," to appear in the *Journal of Guidance, Control, and Dynamics*. Currently available at http://www.mae.cornell.edu/Psiaki/rot_ant_gps_attitude.pdf.

15. Anon., "GP2015 Global Positioning System Receiver RF Front End," *Global Positioning Products Handbook*, GEC Plessey Semiconductors, Wiltshire, U.K., 1996, pp. 49-59.

16. Anon., "TXO4080 Temperature Compensated Crystal Oscillators," Rakon, 1999. http://www.rakon.com/models/browse-model?model_id=89&model_type=O.

17. Spilker, J.J. Jr., "GPS Signal Structure and Theoretical Performance," in *Global Positioning System: Theory and Applications, Vol. I*, Parkinson, B.W. and Spilker, J.J. Jr., eds., American Institute of Aeronautics and Astronautics, (Washington, 1996), pp. 57-119.

18. Stengel, R.F., *Optimal Control and Estimation*, Dover, (New York, 1994), pp. 340-364, 460-488.