JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 25, No. 5, September–October 2002

Automated Tuning of an Extended Kalman Filter Using the Downhill Simplex Algorithm

Thomas D. Powell
The Aerospace Corporation, Los Angeles, California 90009-2957

A method of tuning a Kalman filter by means of the downhill simplex numerical optimization algorithm is presented. The problem is defined by a brief description of the Kalman filter and the extended Kalman filter and the sensitivity of filter performance to process noise and measurement noise covariance matrices Q and R. The filter tuning problem for a system processing simulated data is then formulated as a numerical optimization problem by defining a performance index based on state estimate errors. The resulting performance index is then minimized using the downhill simplex algorithm. The technique is then applied to three numerical examples of increasing complexity to demonstrate its practical utility.

Introduction

SELECTION of process and measurement noise statistics, commonly referred to as "filter tuning," is a major implementation issue for the Kalman filter. This process can have a significant impact on the filter performance. Gelb[1] (Sec. 7.3) describes the sensitivity of the steady-state covariance of a scalar Kalman filter to Kalman gain selection, which illustrates the effect of filter tuning. In practice, Kalman filter tuning is often an ad hoc process involving a considerable amount of trial-and-error to obtain a filter with desirable—qualitative or quantitative—performance characteristics.

Maybeck[2] and others have suggested that a Kalman filter can be tuned with a numerical minimization technique. The concept of tuning by means of numerical optimization allows a digital computer to replace the designer in this tedious, repetitive task, the type of task at which digital computers excel.

The numerical minimization technique applied here is the downhill simplex method, which is a function optimization algorithm available in several programming languages in the *Numerical Recipes*[3] series and which uses only function evaluations—no derivatives—to locate a local minimum of the objective function. Here, the objective function is the root mean square (rms) of the state estimation errors (estimate minus truth), which assumes "true" states are available. This is the case when the filter is applied to simulated data.

The goal of this paper is to provide the interested reader sufficient information to apply the technique to a particular estimation problem. Therefore, emphasis is placed upon a series of illustrative examples of the technique, in addition to its general description.

The paper is organized as follows: Some background information on the Kalman filter, the mathematical assumptions upon which it is based, and the problem of filter tuning are presented first. The following sections describe filter tuning as a numerical optimization problem, followed by a description of the downhill simplex algorithm and some of the implementation issues involved with applying it to filter tuning. The simplex tuning method is then demonstrated using three examples of increasing complexity. The final section discusses the strengths and weaknesses of the technique, with some suggestions on how it can be improved.

Kalman Filter

Before describing the Kalman filter tuning problem, it is helpful to provide an overview of the algorithm to see where the tuning variables arise. There are many references on estimation Kalman filter theory. Maybeck[2] and Gelb[1] are good introductory texts on the subject.

The Kalman filter is the optimal solution to a specific class of state estimation problems. The goal is to estimate a state variable x, usually a vector of multiple states, at some discrete time t_i, given a sequence of measurements y_i. The value of x at time t_i is denoted x_i. For a linear discrete-time system the state variable dynamics are given by

$$x_{i+1} = Ax_i + w_i$$

and the measurement y_i is a linear function of x_i defined by

$$y_i = Hx_i + v_i$$

The linear system is subject to disturbances in the dynamics equation w_i and the measurement equation v_i. These disturbances are often referred to as process and measurement noise, respectively. They are commonly assumed to be zero mean, white, Gaussian random variables with covariances given by $E[w_i w_i^T] = Q$ and $E[v_i v_i^T] = R$, respectively.

The Kalman filter is a solution to this state estimation problem, which can be mathematically proven to be the optimal (minimum variance) estimator under certain assumptions. The two basic assumptions upon which its optimality rests are the linearity of the

Thomas D. Powell is a Project Engineer in the GPS Joint Program Office at The Aerospace Corporation. He received his B.S. in aeronautical and astronautical engineering from Purdue University, his M.S. in aerospace engineering from the University of Texas at Austin, and his Ph.D. in aerospace engineering from the University of California, Los Angeles. He currently supports the development of the new military handheld Digital Advanced GPS Receiver (DAGR) as well as spacecraft applications of GPS. E-mail: thomas.d.powell@aero.org.

Received 19 March 2001; revision received 21 November 2001; accepted for publication 30 April 2002. Copyright © 2002 by Thomas D. Powell. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 0731-5090/02 $10.00 in correspondence with the CCC.

dynamic system in question and that the disturbances corrupting the dynamic and measurement models can be described by white, Gaussian random variables. These assumptions combine to ensure that all variables of interest retain Gaussian distributions because linear combinations of Gaussian random variables remain Gaussian.

The Kalman filter processes each measurement and propagates the state and covariance estimates to the next measurement time using the following time and measurement updates.

Time update:

$$\bar{x}_{i+1} = A\hat{x}_i, \qquad \bar{P}_{i+1} = AP_iA^T + Q$$

Measurement update:

$$\hat{x}_i = \bar{x}_i + K_i(y_i - H\bar{x}_i), \qquad K_i = \bar{P}_iH^T\left[H\bar{P}_iH^T + R\right]^{-1}$$

$$P_i = (I - K_iH)\bar{P}_i$$

The Kalman gain K_i in the state measurement update equation relates the relative weight given to the filter's internal dynamic model and the measurement information. A larger K_i places more weight on the measurements, hence "de-weighting" the filter's internal dynamic model. A smaller value of K_i places less weight on the measurement information and more weight on the filter's internally predicted state. As such, the Kalman gain K_i controls how new measurement information is combined with the internal dynamic model and so determines the performance of the filter. Examining the preceding equations for the filter time and measurement update, the Kalman gain K_i clearly depends on the state and measurement noise covariance matrices Q and R. Therefore, the choices of Q and R directly affect the performance of the Kalman filter.

Nonlinear Systems and the Extended Kalman Filter

Real-world dynamic systems obey physical laws that are generally much more complex than the simple linear models used to derive the optimal Kalman filter. The complexity, or nonlinearity, of these real-world systems invalidates the assumptions of linearity and Gaussian random variables upon which the optimal Kalman filter is based.

If the dynamic system of interest is nonlinear, the standard approach is to linearize the system by reformulating the problem in terms of a perturbation from some reference nonlinear trajectory. If this perturbation variable is close enough to the reference trajectory, then it can be assumed to obey a linear dynamic model. The Kalman filter is then applied to this perturbation state, although the optimality of the estimates can no longer be guaranteed because the disturbances can no longer be assumed to be Gaussian. Although the Kalman filter cannot be proven to be optimal for nonlinear systems, the Kalman filter is applied to the linearized system. This extension of the Kalman filter to the linearized nonlinear problem is commonly referred to as the extended Kalman filter (EKF).

Modeling a physical system requires making assumptions about the type and order of mathematical model to be used. When a nonlinear dynamic system is linearized about some reference trajectory, higher-order nonlinear terms in the dynamics are discarded. The resulting model is always imperfect and so introduces modeling errors into the estimation problem.

The only way the filter can account for unmodeled dynamic or measurement effects is to somehow account for them in the process and measurement "noise" covariance, whether they are truly noise-like or not. The internal modeling of these terms as Gaussian noise is often grossly inaccurate, but the tuning process can mitigate some of the shortcomings of this model.

Kalman Filter Tuning

Kalman filter tuning refers to the process of selecting the elements of the process noise covariance Q and measurement noise covariance R matrices to improve the filter's state estimates with respect to some performance measure. If the statistics of the process and measurement noise process models in the filter do not "match" the actual statistics of these noise processes, the filter estimates will be degraded.

The choices of Q and R are the primary factors that determine the Kalman gain K_i, which in turn determines the operation of the Kalman filter. Most real applications are concerned with the steady-state behavior of the filter, when the Kalman gains are relatively constant. A filter designer could therefore tune the Kalman filter by selecting the elements of the steady-state Kalman gain directly. This is often done in practice, although this approach is not examined here.

How the Kalman filter is tuned will depend on how it is used. The filter can operate on real data or simulated data. It can process this data in real time, or it can postprocess some historical data. If simulated data are used, the true states that were used to generate the data are known and can be used to tune the filter by minimizing the errors between the estimates and the known truth. This is the method described here.

However, the true states are never known in a real application. In this case the filter measurement residuals can be used to tune the filter, either by minimizing the rms of the residuals over time or by computing an autocorrelation function of the residuals. Kalman filter theory states that for a linear system the filter residuals will be uncorrelated with time (that is, "white") if all useful information has been extracted from the measurements. The filter could be tuned using this idea by constructing a performance index based on the statistical whiteness of the residuals.

Tuning a filter for a real time, real data application is the most challenging problem. The conventional approach is to design and test the filter using some simulation of the operational environment, which is as accurate as possible. A filter tuned with this approach should yield acceptable performance if the actual operating environment can be sufficiently modeled, although unpredictable disturbances often arise that invalidate the underlying assumptions. In the end the designer tunes the filter based on the best available knowledge of the operating environment. Continuous improvements to the models and tuning can also be made as real-world operational experience and data are accumulated.

The measurement noise statistics can often be determined through sensor testing and calibration. It is also possible to extract measurement noise statistics from the measurement residuals.[4] For these reasons the measurement noise covariance R is often assumed to be known and can be removed from the tuning process.

In general terms, a "larger" value of the Q matrix will have the same effect as a "smaller" value of R, and vice versa. Keeping in mind that Q and R are matrices and that in general one cannot "divide" Q by R, it is the notion of the ratio Q/R that determines the filter gains. This idea is often used to simplify the tuning process by fixing R and selecting elements of Q only.

In general, it is difficult to determine the effect of multiple changes to the tuning parameters, so that an analyst will often tune each parameter individually, holding the others constant. This parameter-by-parameter tuning can require a very large number of manual trials. When the dynamics are nonlinear and the number of parameters begin to grow to more than two or three, the trial-and-error technique can become quite tedious and time consuming. Furthermore, a grid search technique can require an unreasonably large number of simulation runs to sufficiently cover the tuning parameter space.

Filter Tuning as a Numerical Optimization Problem

Given the difficulty of tuning a Kalman filter through trial and error, it is desirable to convert the tuning problem to a more tractable numerical optimization problem. This would allow the application of a hands-off numerical optimization technique, which would exploit the primary benefit of digital computers in performing repetitive, tedious computations.

The primary benefit of the method described here is that it is a hands-off approach, designed to save time and effort for the filter designer. No claims can be made about the speed of the method. However, as long as this method may take to select a set of tuning parameters, the fact is that the computer, not the filter designer, is tuning the filter, freeing the analyst from this tedium.

Performance Index Definition

To apply a numerical optimization algorithm to tuning the Kalman filter, the tuning problem must be expressed as a numerical optimization or function minimization problem. This means constructing a scalar objective function J, which is to be minimized as a function of some set of N independent variables. Assuming that the parameters to be selected in the tuning process are the elements of the filter's process noise covariance matrix Q, the filter performance index could be written as $J = J(q_1, q_2, \ldots, q_{N_Q})$, where N_Q is the number of elements of the matrix Q to be selected.

The performance index could be based on a number of filter variables and their values, which are available during computer simulation. The performance index used in the examples presented here, as described later, is based on the rms of the state estimation errors:

$$J_k(q_{11}, q_{22}) = \left[\frac{1}{N}\sum_{t_i = t_0}^{t_f}\left(\bar{e}_i^T W \bar{e}_i\right)\right]^{\frac{1}{2}}$$

where $\bar{e}_i^T = [(\hat{x}_{1_i} - x_{1_i}), (\hat{x}_{2_i} - x_{2_i})]^T$ is the vector of state estimation errors, W is a state weighting matrix, t_0 is the initial discrete time of the tuning interval, t_f is the final discrete time of the tuning interval, and N is the number of data points in the tuning interval.

For linear systems the measurement residuals should be white, and a performance index that tests this property of the residuals could be implemented. However, this type of performance index would not be suited for nonlinear systems, where whiteness of the residuals is not guaranteed. A performance index based on the rms of the measurement residual could also be used, although small measurement residuals do not guarantee small state estimation errors, particularly for systems where bias parameters are to be estimated.

A state weighting matrix W can be chosen to weight individual states differently in a performance index based upon the least squares of the state estimation errors. In general, different weighting matrices should lead to different tuning solutions for nonlinear problems. For truly linear problems the weighting matrix W will alter the shape of the performance index function, but the location of the minimum should be constant.

Evaluating this performance index J entails processing some data set, either real or simulated, to produce a set of state estimates and filter residuals. Depending on the complexity of the simulation, this function evaluation could be computationally expensive. Monte Carlo simulation can be used to compute J, although it is not necessary for the simplex tuning method and might be impractical for computationally expensive problems.

Some Candidate Optimization Algorithms

The most simple tuning optimization technique next to trial and error is to evaluate each tuning parameter over a range of values, thereby performing a grid search of the tuning parameter space. This method is very inefficient and might not yield acceptable filter performance if the grid is not located near the best tuning parameter solution.

Gradient methods of optimization require the knowledge of the partial derivatives of the function to be minimized with respect to the independent variables. The derivatives can be computed analytically if the mathematical expression for the function is known, or by numerical differentiation if it is not. These methods also offer the potential for proof of optimality through conditions on the second-order derivatives of the function. However, gradient methods are not well suited for functions that are discontinuous and "noisy" in nature, as numerical differentiation usually fails under these conditions.

Simulated annealing (SA), also described in *Numerical Recipes*,[3] is a numerical minimization technique similar to the downhill simplex algorithm, but which emulates the slow cooling—annealing—of a liquid material in its movements through the parameter space. However, Press et al.[3] describe the application of SA to problems with large numbers of discretely valued independent variables, as arise in circuit element placement problems, or the classic traveling salesman problem. Because the filter tuning problem involves continuously valued independent variables and because the performance index for a filter tuning problem can be computationally expensive, SA is not examined here.

Genetic algorithms (GA) are shown to have utility for tuning a Kalman filter in Ref. 5. This work also proposes a performance index that includes a consistency check between the Kalman filter state error estimates and the filter covariance.

Downhill Simplex Algorithm

The downhill simplex method, as discussed in Ref. 3, is a numerical optimization algorithm that minimizes a scalar objective function of N independent variables, using function evaluations only. It does not require derivatives of the function being minimized.

A simplex, as defined in Ref. 6, is a collection of $N + 1$ points in an N-dimensional space and all their interconnecting line segments. In two dimensions a simplex is a triangle. In three dimensions it is a tetrahedron. The function to be minimized is evaluated at each of the $N + 1$ simplex vertices.

The downhill simplex algorithm is sometimes referred to as the "Amoeba" routine, its motion likened to a fictitious blob "oozing" into the valleys of the performance index in search of the minimum value. The simplex algorithm attempts to locate a minimum of the function by a series of movements in the N-dimensional space. These movements consist of three basic types: reflection, expansion, and contraction. The details of these basic movements are fully described in Ref. 6, but a brief overview is given here.

The algorithm takes the point in the simplex with the highest function value and "reflects" it along a line formed by the point and the centroid of the remaining points to some point on the "far" side of the centroid. If a decrease in the function is encountered at this new point, the simplex will "expand" further in that direction. If an increase in the function is found, the simplex will "contract" away from it. The distance the point is moved in these expansions and contractions is determined by a set of constant expansion and contraction coefficients. In this way the simplex moves through the N-dimensional topology of the function's range space. If it reaches a point where one of its points falls into a "hole," the entire simplex contracts onto the point with the lowest value and is drawn "down" towards the point with the minimum value. Convergence is defined in terms of the relative values of the function at each of the vertices. A tolerance is set and convergence is achieved when the function values for all points on the simplex fall within an interval of that size.

Implementation Issues

There are number of implementation issues that must be resolved in order to apply the automated simplex tuning approach. All of these issues are problem-specific and will depend to a large degree on the subjective judgment of the filter designer.

Automated Tuning Software Architecture

To apply the simplex method to filter tuning, it is necessary to convert the Kalman filter simulation to an equivalent function, which returns a scalar value of a filter performance index. The implementation of this method requires the Kalman filter function to be a stand-alone software module that can be passed the values of the independent variables—the tuning parameters—by the simplex algorithm and will return the scalar performance index. The exact code implementation will be language-dependent. Figure 1 illustrates this simplex tuning algorithm.

Parameterization of the Tuning Problem

The filter to be tuned must be parameterized for the numerical optimization problem. This entails choosing which parameters are to be used as independent variables for the tuning algorithm, keeping in mind that an $(N \times N)$ matrix Q has $(N + 1) \cdot N/2$ unique elements. In the third example presented here, $N = 7$, and so Q contains $(7 + 1) \cdot \frac{7}{2} = 28$ unique elements. Selecting all 28 unique elements of this Q matrix would be impractical numerically, as the parameter space is very large. Faced with this fact, and the fact that the off-diagonal elements of Q represent cross correlations that are difficult to interpret, the filter designer typically selects only the diagonal elements of Q in tuning the filter. This reduces N_Q from 28 to 7 for this (7×7) example.

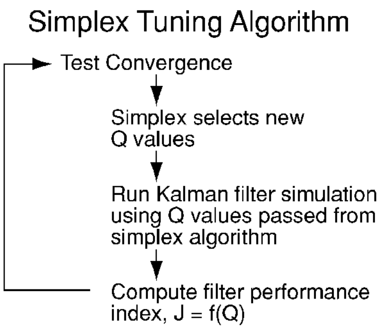
Simplex Tuning Algorithm
Test Convergence
Simplex selects new Q values
Run Kalman filter simulation using Q values passed from simplex algorithm
Compute filter performance index, J = f(Q)

Fig. 1 Simplex tuning outline.

Performance Index Weighting Matrix

The choice of weightings for the filter performance index is also unique to every tuning problem. Although this weighting matrix could be added to the state vector and estimated as another system parameter, this increases the dimension and computational burden of the estimation and tuning problem and is probably best left to the engineering judgment of the filter designer.

Tuning Exclusion of Filter Transient Response from Performance Index

The transient response of the Kalman filter can produce large state estimation errors and depends upon the initial state estimates and state error covariance assumptions. As most applications are concerned with the steady-state behavior of the Kalman filter, the filter state errors during the transient response can be omitted from the filter performance index computation by the choice of t_0 and t_f. The bounds of the tuning interval will depend on the problem at hand and the judgment of the filter designer. This also allows the tuning to emphasize some other phase of the filter operation. By adjusting the endpoints of the tuning interval, emphasis can be placed upon any particular phase of operation, such as the steady-state operation of the filter. This is illustrated in example 3, where the transient response of the filter is excluded from the cost function.

Kalman Filter Stability During Automated Tuning

There are certain numerical conditions required for the Kalman filter to maintain numerical stability. The open-loop stability of the dynamic system, is determined by the eigenvalues of the A matrix. Because the noise covariance matrices $Q = E[w_i w_i^T]$ and $R = E[v_i v_i^T]$ are modeled as expected values of quadratic forms, they are required to be positive definite matrices to guarantee stability—boundedness—of the underlying matrix Riccati equation. The closed-loop stability of the Kalman filter is also dependent on the choice of Q and R and is more important than open-loop stability for state estimation because one can construct a stable Kalman filter for an unstable dynamic system. However, the simplex algorithm is unaware of these stability conditions, and in general there are no constraints to prevent it from selecting tuning parameters that yield an unstable filter.

If a stable filter is the desired outcome of the tuning process, an automated tuning algorithm must eventually achieve this goal to be useful for filter design. Fortunately, a filter tuning performance index based on simulated state estimation errors (estimate minus truth) serves as an effective constraint against divergent filters. A divergent filter will yield arbitrarily large simulated state estimation errors that will in turn be reflected in the tuning performance index J. Experience with this tuning method shows that it does occasionally select divergent tunings, but these occur early in the tuning process and are quickly weeded out.

However, there are some simple ways to preserve the positive definiteness of the Q and R matrices. A penalty function, which assigns the cost function an extremely large value when a nonpositive definite Q or R matrix is evaluated, can keep the parameters in the positive definite region. Selecting the elements of the matrix square roots of Q and R will also maintain positive definiteness.

The method employed here is to perform a change in variables from q_{11} and q_{22} to $\ell_{11} = \log(q_{11})$ and $\ell_{22} = \log(q_{22})$. The logs of the tuning parameters ℓ_{11} and ℓ_{22} are then passed to the simplex algorithm, which searches in the ℓ_{11}, ℓ_{22} parameter space for the minimum performance index. The logs are passed to the Kalman filter routine, which reverses the change of variables $q_{11} = 10^{\ell_{11}}$ and $q_{22} = 10^{\ell_{22}}$ before processing the data.

Number of Monte Carlo Samples

The performance indices for the first two examples shown here are computed over 100 Monte Carlo samples. Although this yields acceptable performance of the tuning algorithm for these simple examples, the minimum number of Monte Carlo samples required for acceptable tuning results is problem-specific. This is true for any type of Monte Carlo simulation analysis in general, and ultimately comes down to engineering judgment. Some factors to be considered are the order of the system, transient response properties of the filter, and computational burden of the simulation. The designer might also wish to obtain a quick "coarse" tuning of the filter with a smaller number of Monte Carlo samples for initial analysis and increase the number of samples for the final "fine" tuning.

If the system has a long, computationally expensive transient response, fewer Monte Carlo samples of greater duration might prove more computationally efficient than more shorter samples. This is the case in example 3 presented later, where a single Monte Carlo sample is used with acceptable results.

The ergodicity of the system in question is also an important consideration that might be difficult to determine for a highly correlated, nonlinear system. Does the average over many samples match the average over long times? Ergodicity is also a problem-specific issue.

Avoiding Nonoptimal Local Minima

Given that the simplex tuning technique employs a numerical optimization technique that uses no derivative information of the performance index, no global optimality statements can be made once the algorithm has converged to a minimum. So the result of the tuning process is not "optimal" in the strict mathematical sense, and a true global minimum cannot be assured. However, confidence in the tuning result can be increased. To avoid converging to local minima, the algorithm can be initialized from several sets of initial conditions to verify that it converges to the same solution. A filter designer might find the converged tuning parameters acceptable and place more value on the time saved by this tuning technique than on the assurance of global optimality.

Tuning Example 1: Simplex Tuning of the Linear Case

The simplex tuning algorithm is now applied to a simple discrete-time linear system. Denoting discrete time t_i with the subscript $(\cdot)_i$, the example discrete-time linear system dynamics and measurements are defined by the following state-space model:

$$\begin{bmatrix} x_{1_{i+1}} \\ x_{2_{i+1}} \end{bmatrix} = \begin{bmatrix} 0.9 & 1 \\ 0 & 0.8 \end{bmatrix} \begin{bmatrix} x_{1_i} \\ x_{2_i} \end{bmatrix} + \begin{bmatrix} w_{1_i} \\ w_{2_i} \end{bmatrix}$$

$$y_i = [1 \quad 0] \begin{bmatrix} x_{1_i} \\ x_{2_i} \end{bmatrix} + v_i$$

for which the A and H matrices are defined as

$$A = \begin{bmatrix} 0.9 & 1 \\ 0 & 0.8 \end{bmatrix}, \qquad H = [1 \quad 0]$$

Simulated data are generated using a Gaussian pseudorandom number algorithm and the following statistics:

$$Q = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad R = 1$$

The measurement sequence y_i generated by this true model is processed with a Kalman filter for which the value of Q used in the filter algorithm is varied. Let the value of Q used in the Kalman filter be denoted by

$$\hat{Q} = \begin{bmatrix} q_{11} & 0 \\ 0 & q_{22} \end{bmatrix}$$

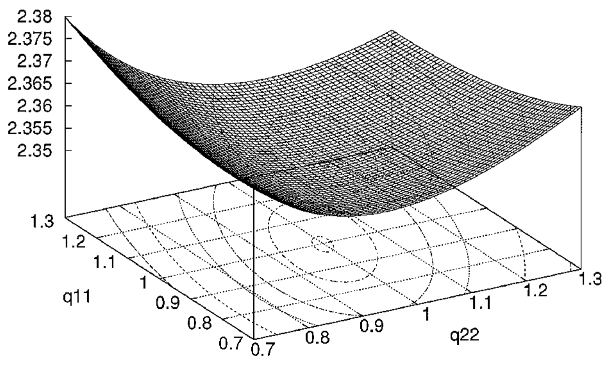

Fig. 2 Monte Carlo cost function for linear 2 state system.

and assume that the filter uses the correct value of $\hat{R} = R = 1$. The filter also uses the correct value of the dynamic matrix A, which has both eigenvalues within the unit circle. Therefore, if $\hat{Q}$ chosen by the tuning algorithm and R are both positive definite the dynamic system and Kalman filter will be asymptotically stable.

In the linear case, where the process and measurement noise processes are actually Gaussian—a hypothetical case—tuning the Kalman filter consists of matching the filter's Q and R to the actual noise statistics. The effect of incorrect tuning is demonstrated by varying the values of q_{11} and q_{22} away from their true values used to generate the data. The combination of q_{11} and q_{22} can be treated as a point in a two-dimensional parameter space, and the filter performance index can be plotted as a surface $J = J(q_{11}, q_{22})$.

The Monte Carlo performance index J_k for the first two examples is computed for each of 100 Monte Carlo samples ($N_{\mathrm{MC}} = 100$), and then the overall performance index J is the average value of J_k over the N_{MC} samples. The value of J_k for each Monte Carlo sample is the sum of the rms state estimation errors for each state:

$$J = \frac{1}{N_{\mathrm{MC}}} \sum_{k=1}^{N_{\mathrm{MC}}} J_k(q_{11}, q_{22})$$

$$= \frac{1}{N_{\mathrm{MC}}} \sum_{k=1}^{N_{\mathrm{MC}}} \left\{ \left[\frac{1}{N} \sum_{t_i = t_0}^{t_f} \left(\bar{e}_i^T W \bar{e}_i \right) \right]_k^{\frac{1}{2}} \right\}$$

where the state weighting matrix $W = I$. State errors outside of the tuning interval are not included in the performance index.

Figure 2 plots the surface of the performance index $J(q_{11}, q_{22})$ vs its two independent variables q_{11} and q_{22}. The surface was constructed by evaluating the filter performance index over a grid of points evenly spaced over the intervals $[0.7 \le q_{11} \le 1.3]$ and $[0.7 \le q_{22} \le 1.3]$. Contours of constant performance index value, or level curves, are drawn on the function surface and projected onto the (q_{11}, q_{22}) plane below the surface. The projected contours form concentric curves around the "valley" centered at $(q_{11} = 1, q_{22} = 1)$. The minimum of the performance index can be seen from the diagram to occur at $(q_{11} = 1, q_{22} = 1)$, which is exactly where it would be expected to be if the data were generated with true process noise statistics given by $Q = I$.

Because there are two independent variables, q_{11} and q_{22}, $N = 2$, and the simplex for this case will consist of $N + 1 = 3$ points in the two-dimensional parameter space.

In general, it is best to distribute the initial points of the simplex widely within the range of the independent variables. In this case the true Q is known, but this is ignored, and the three initial points of the simplex are chosen to be the following:

1) Point 1 is $q_{11} = 30$, $q_{22} = 30$.
2) Point 2 is $q_{11} = 30$, $q_{22} = 1$.
3) Point 3 is $q_{11} = 1$, $q_{22} = 30$.

The filter performance index for the linear example shown in Fig. 1 can also be represented by the topographic projection of its level curves onto the q_{11}, q_{22} plane.

If the q_{11}, q_{22} pairs evaluated by the simplex algorithm are superimposed on this topographic plot, the tuning trajectory of the simplex algorithm can be observed. This tuning trajectory for the linear case is shown in Fig. 3. Because the true process noise statistics are known to be $Q = I$, convergence of the simplex tuning algorithm to the point $(q_{11} = 1, q_{22} = 1)$ is the desired outcome. Figure 3 plots the successive positions of the "low" vertex—the vertex with the lowest function value of the three—as the simplex algorithm tunes the Kalman filter for this case. Although it is difficult to represent the time history of the points in a static chart, the arrows describe the general motion of the low point toward the function minimum.

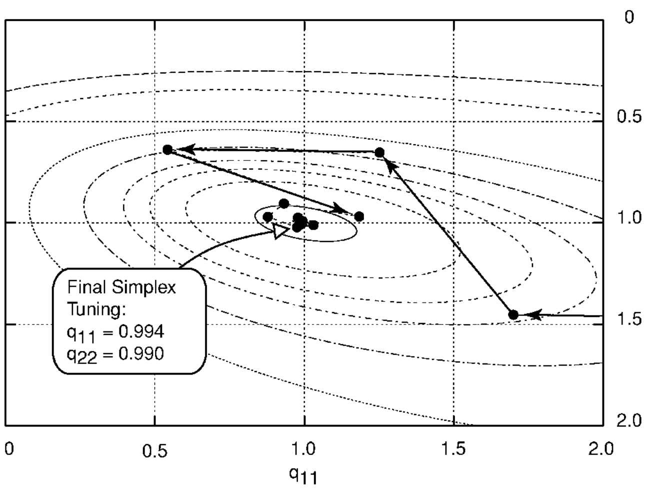

Fig. 3 Tuning parameter trajectory for example 1: Linear two-dimensional system.

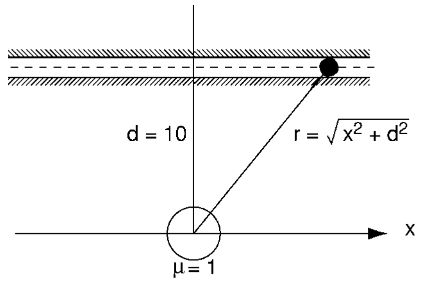

Fig. 4 Two-state nonlinear dynamic system.

Keeping in mind that the simplex for this example is a set of three (q_{11}, q_{22}) pairs, the final tuning parameters must be selected from the final set of three simplex points:

1) Point 1 is $q_{11} = 0.989606$, $q_{22} = 1.005817$, $J = 2.354177$.
2) Point 2 is $q_{11} = 1.010932$, $q_{22} = 1.004144$, $J = 2.354171$.
3) Point 3 is $q_{11} = 0.994039$, $q_{22} = 0.990321$, $J = 2.354170$.

Here, the final simplex point 3 yields the lowest filter performance index.

The simplex tuning algorithm was also applied to the following neutrally stable discrete-time linear system:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 0.9 \end{bmatrix}, \qquad H = [1 \quad 0]$$

The system is neutrally stable because one of the eigenvalues of the A matrix lies on the unit circle. When the simplex tuning algorithm was applied to this system with the same pseudorandom noise processes as before, the result was the same. The simplex tuning algorithm successfully selected the true process noise covariance matrix Q as before.

Tuning Example 2: EKF Applied to a Two-Dimensional Nonlinear System

The second example is chosen to illustrate some of the real-world issues that complicate Kalman filter design and tuning: nonlinearity and non-Gaussian disturbances. The two-dimensional state-space system chosen is based on the one-dimensional central force motion of a particle, as illustrated in Fig. 4. The second-order nonlinear differential equation of motion for the system is

$$\ddot{x} = -\mu x / r^3$$

and the nonlinear measurement is given by

$$y = r = \sqrt{x^2 + d^2} = h(x_1, x_2)$$

The two state variables are defined as $x_1 = x$ and $x_2 = \dot{x}$, and the corresponding nonlinear state-space equations are

$$\dot{x}_1 = f_1(x_1, x_2) = x_2, \qquad \dot{x}_2 = f_2(x_1, x_2) = \frac{-\mu x_1}{\left(x_1^2 + d^2\right)^{\frac{3}{2}}}$$

These nonlinear dynamic and measurement equations must be linearized in order to apply the EKF to this problem. The linearizations are defined by

$$A = \left.\frac{\partial f}{\partial x}\right|_{x_i^*}, \qquad H = \left.\frac{\partial h}{\partial x}\right|_{x_i^*}$$

These partial derivatives are evaluated at the point x_i^* along a nonlinear reference trajectory. This reference trajectory is integrated numerically and is reinitialized after each measurement update to the most recent a priori state estimate $\hat{x}_i$.

The discrete-time EKF is then formulated using the state transition matrix $\Phi(t_j, t_i)$, which is the solution to the differential equation

$$\dot{\Phi}(t_j, t_i) = A\Phi(t_j, t_i)$$

with the boundary condition $\Phi(t_i, t_i) = I$. The elements of the matrix A are defined by $\{a_{ij}\} = \partial\dot{x}_i/\partial x_j$. Therefore, for this example,

$$A = \begin{bmatrix} 0 & 1 \\ \mu\left(2x_1^2 - d^2\right)/\left(x_1^2 + d^2\right)^{\frac{5}{2}} & 0 \end{bmatrix}$$

and the state transition matrix is computed at each time step using fourth-order Runge–Kutta numerical integration.

To demonstrate the process of tuning an extended Kalman filter under more unfavorable conditions, the system is corrupted by a multiplicative disturbance with a χ^2 probability distribution as well as an unmodeled bias produced by a second-order Markov process.

The second example was constructed in this way so that the true noise statistics are unknown. Clearly, the system is not only nonlinear, but subject to non-Gaussian disturbances. What tuning parameters will yield the best extended Kalman filter performance are unknown prior to the tuning process.

Because the true tuning parameters for this problem are unknown, the neighborhood of the minimum cost in the q_{11}, q_{22} parameter space is located using a coarse grid search for the purpose of plotting. The surface plot of the cost function vs q_{11} and q_{22} for this nonlinear example is shown in Fig. 5. The topography of this function reveals a minimum near the point $q_{11} = q_{22} = 1.5$. The test of the simplex tuning algorithm here is if it converges to this apparent minimum of the cost function.

The trajectory of the low vertices of the three vertex simplex for the second example is shown in Fig. 6. Once again, the simplex tuning algorithm has located the minimum of the filter performance index for this nonlinear, non-Gaussian estimation problem.

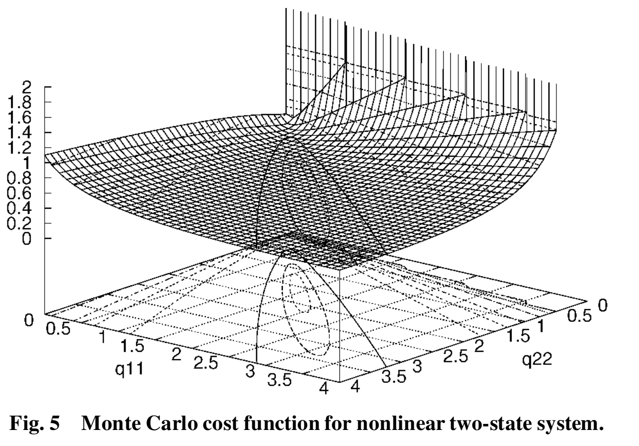

Fig. 5 Monte Carlo cost function for nonlinear two-state system.

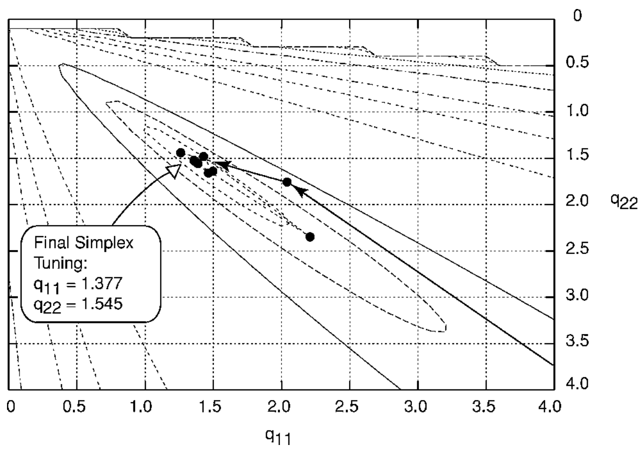

Fig. 6 Tuning parameter trajectory for example 2: Nonlinear two-dimensional system.

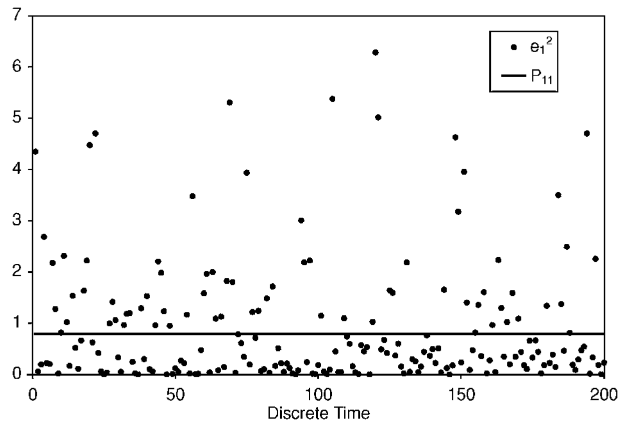

Fig. 7 Sample of filter state estimation errors and covariance for linear example 1.

The final converged set of three (q_{11}, q_{22}) pairs for the two-dimensional nonlinear example are as follows:

1) Point 1 is $q_{11} = 1.386538, q_{22} = 1.556232, J = 0.47820887$.

2) Point 2 is $q_{11} = 1.377049, q_{22} = 1.544843, J = 0.47820885$.

3) Point 3 is $q_{11} = 1.375649, q_{22} = 1.545644, J = 0.47820905$.

The minimum performance index occurs at simplex point 2 for this case.

In Kalman filter theory the state error covariance is defined as the expected value of squared state estimation errors

$$P = E[(x - \hat{x})(x - \hat{x})^T]$$

For linear systems driven by Gaussian noise, the filter covariance will indeed represent the expected value of the squared state estimation errors.

However, for nonlinear systems no statements can be made about the relationship of the filter covariance and the statistics of the state estimation errors. The filter covariance P may indeed accurately model the state estimation errors for a given nonlinear problem. However, there might be filters applied to nonlinear systems for which this is not the case, although these filters can give acceptable performance for their applications.

This can be illustrated by plotting the estimation error (estimate minus truth) for the first (position) state and associated filter covariance for the two-dimensional linear and nonlinear examples just presented. The state error and covariance for the linear example 1 are shown in Fig. 7, and the same variables for the nonlinear example 2 are shown in Fig. 8.

Examining the plots for the linear case shows that the errors are well described by the filter covariance. However, this is not the case for the nonlinear case, despite the fact that the errors shown were produced with the best tuning from the simplex tuning algorithm,

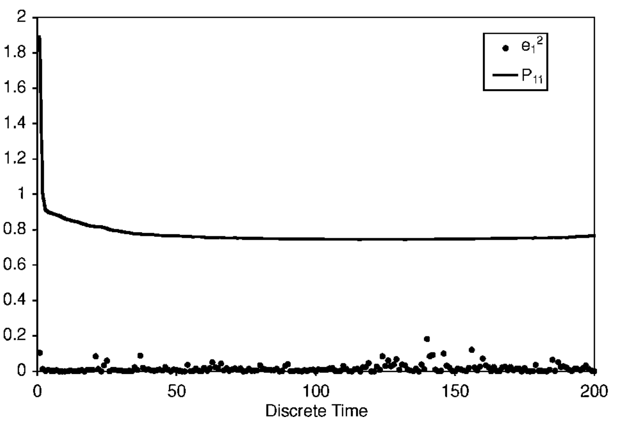

Fig. 8 Sample of filter state estimation errors and covariance for nonlinear example 2.

based on the state error performance index J. This illustrates that the tuning which results in minimum state errors for the nonlinear problem did not result in a filter covariance that accurately represented the expected value of the squared errors. But this is consistent with the theory, which makes no assurances that the covariance will represent the errors for nonlinear problems.

A performance index based upon the statistical consistency of the filter covariance with the state estimation errors is developed in Ref. 5. The consistency check is used to constrain the tuning search to filters for which the covariance matches the statistics of the state estimation errors. Although this is a desirable property for a filter and should be expected for linear systems, a filter covariance that matches the statistics of the state estimation errors does not guarantee the minimum state estimation errors for nonlinear systems, as illustrated by these results.

Tuning Example 3: Autonomous Geosynchronous Satellite Navigation Using Global Positioning System

The final example is the tuning of an EKF processing global positioning system (GPS) pseudorange measurements to perform orbit determination for a spacecraft in a geosynchronous (GEO) orbit. The GEO navigation problem imposes some unique constraints on the GPS navigation problem.

The primary problem for navigation of GEO spacecraft using GPS is that the visibility of the GPS spacecraft from GEO is very limited because the GPS spacecraft are below the GEO spacecraft and the GPS broadcast antenna is pointed at the surface of the Earth, away from the GEO spacecraft above. As a result, the only signals that arrive at GEO are those which spill past the limb of the Earth.[7]

The orbit of the spacecraft is perturbed by higher-order Earth geopotential terms, solar radiation pressure, and lunar and solar gravity. The result is clearly a nonlinear dynamic system. The measurements are corrupted by white noise in addition to GPS ephemeris and clock errors.

The vehicle state vector includes six stabilized Keplerian orbital elements $\{\alpha, \beta, \gamma, a, \Omega, i\}$, and two elements for the GPS receiver's clock phase and frequency error estimates $\{\phi, \omega\}$. Although clock phase is normally expressed in units of seconds, it is often represented in the filter in distance units by multiplying by the speed of light. Similarly, clock frequency can be represented in units of velocity. In addition, the state vector is appended with a pass bias b for each of n GPS spacecraft in view $\{b_1, \ldots, b_n\}$, which are intended to account for the unmodeled GPS broadcast ephemeris and clock errors.

The general form of the state dynamics, including the appended clock and pass bias states, is given by

$$\begin{aligned}\dot{\bar{X}} &= [\dot{\alpha}, \dot{\beta}, \dot{\gamma}, \dot{a}, \dot{\Omega}, \dot{i}, \dot{\phi}, \dot{\omega}, \dot{b}_1, \ldots, \dot{b}_n]^T \\ &= [f_\alpha(\bar{X}), f_\beta(\bar{X}), f_\gamma(\bar{X}), f_a(\bar{X}), f_\Omega(\bar{X}), f_i(\bar{X}), \omega, 0, 0, \ldots, 0]T\end{aligned}$$

where each function $f_{(\cdot)}(\bar{X})$ is some nonlinear function of the state vector $\bar{X}$.

The scalar pseudorange measurement equation for GPS satellite j is modeled in the filter as

$$y_j = \left\| \bar{R}_{\mathrm{GPS}_j} - \bar{R}(\bar{X}) \right\| + \phi + b_j$$

where $\bar{R}_{\mathrm{GPS}_j}$ is the Cartesian position vector for GPS satellite j, which is assumed to be known; $\bar{R}(\bar{X})$ is the Cartesian position of the GEO satellite, which is a function of the state vector $\bar{X}$; ϕ is the GPS receiver clock phase offset to be estimated; and b_j is the pass bias for GPS satellite j, which accounts for other unmodeled biases. Both of these biases are expressed in distance units for consistency within the pseudorange measurement.

The seven process noise covariance parameters to be selected by the simplex algorithm for this problem are the six associated with the stabilized Keplerian elements $\{q_\alpha, q_\beta, q_\gamma, q_a, q_\Omega, q_i\}$ and the process noise to be applied to the pass bias state for each GPS satellite in view q_b. It is assumed that the receiver clock noise statistics have been determined through calibration, and so the clock state process noise parameters $\{q_\phi, q_\tau\}$ are not tuned by the simplex algorithm.

The EKF for this problem therefore has a minimum of eight state vector elements, with as many as 12 or 13 depending on the visibility to the 24 GPS satellites. The additional pass bias states are added and removed to the state vector as GPS satellites come into and out of view. If the process noise covariance matrix Q is chosen to be diagonal, tuning the EKF could require choosing up to 13 separate diagonal elements when GPS visibility is high. To simplify the tuning somewhat, a single q value is used for all of the GPS pass biases, which reduces the q parameter space to only nine variables. The GPS receiver clock error statistics can be assumed, which fixes the q values for these two states. This leaves seven q values to select in order to tune the EKF: $\{q_\alpha, q_\beta, q_\gamma, q_a, q_\Omega, q_i, q_b\}$.

This is a typical tuning problem faced by a filter designer or analyst. How to select seven q values to achieve the best performance from this EKF? There may be some intuition from the dynamics to guide the filter tuning, but the tuning of this EKF will often involve some amount of trial and error.

The Monte Carlo performance index J_k for the GEO GPS navigation problem is simply the rms three-dimensional position error of the magnitude of the difference between true Cartesian position and the filter estimate:

$$J_k(Q) = \left\{ \frac{1}{N} \sum_{t_i = t_0}^{t_f} [\bar{R}(\hat{\bar{X}}_i) - \bar{R}(\bar{X}_i)]^T [\bar{R}(\hat{\bar{X}}_i) - \bar{R}(\bar{X}_i)] \right\}^{\frac{1}{2}}$$

where $\hat{\bar{R}}_i$ is the filter estimate of the GEO satellite cartesian position; $\bar{R}_i$ is the true Cartesian position; and t_0, t_f, and N are defined as before. For the GEO problem $t_0 = 2$ days to exclude the transient response of the filter from the performance index and $t_f = 5$ days. The filter was run with a 15-min measurement interval.

Unlike the preceding two-dimensional examples, the EKF for the GEO application was not tuned using a Monte Carlo based performance index. The simplex tuning algorithm used a single simulation run. This is primarily because of the increased complexity and computational burden of this problem. The resulting tuning parameters, when used with different simulation samples, give consistent performance, so that the effect of not using a Monte Carlo based cost function was small in this case.

Obviously, with seven independent variables a plot of the cost function vs Q matrix elements cannot be illustrated in two-dimensional picture. What can be plotted, though, is the value of the filter performance index for each set of Q elements selected by the simplex tuning algorithm. This plot, shown in Fig. 9, shows the progress of the simplex tuning and the eventual convergence of the algorithm as successive iterations improve the performance index from several hundreds of meters to approximately 17 m rss position accuracy.

The simplex tuning algorithm proved to be invaluable for evaluating the navigation performance of a GEO satellite in Ref. 7. A total of 15 different GPS antenna configuration and GEO user GPS receiver clock quality combinations were evaluated. To assess the navigation accuracy for each case, each case needed to be tuned

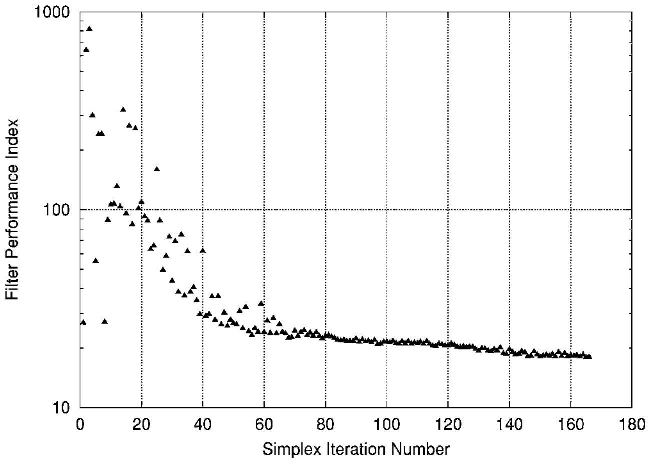

Fig. 9 Filter cost function vs simplex iteration number for GEO spacecraft navigation example.

independently. Problems of this type best demonstrate the utility of this method of filter tuning.

Conclusions

The results demonstrate that the Downhill simplex technique has great utility for tuning the Kalman filter for linear and nonlinear applications. It can be applied to simulated data, or to real data when a highly accurate "truth" reference is available, as is often the case in postprocessing. The technique is applied here to three example tuning problems, and it should extend well to other similarly defined filtering applications.

Tuning is accomplished by constructing a filter performance index based on state estimation errors and coding the Kalman filter simulation of interest as a stand-alone function that can be passed as tuning parameters of independent variables. This function then returns the value of the scalar filter performance index as a function of these tuning parameters. The technique exhibits rapid convergence to the minimum performance index from a variety of initial conditions. This repeatable convergence is a useful trait for an automated tuning technique.

The technique is demonstrated here on simulated data, for which the true states are known. This is a situation that often arises when a filtering algorithm is being designed and tested in a controlled setting prior to actual field testing with real-world data. The technique could also be applied to real data if an independent truth reference is available.

Although the downhill simplex method was used here, other numerical optimization techniques might be suitable for filter tuning. Given the noisy nature of the data, a method that employs function evaluations only is desirable over techniques which require analytic or numerical gradient information.

The cost function used here was a simple rms of the state estimation errors taken over time for a number of Monte Carlo simulation samples. Other filter performance indices based on the magnitude or autocorrelation of the measurement residuals could also be examined.

Although this method has shown great utility for tuning an EKF, the output of the filter should be closely examined to verify that it is performing as expected. When simulated truth data are available, the state estimation errors should be compared with filter error covariance for consistency.

Furthermore, by allowing the rapid evaluation of different Kalman filter configurations this technique allows the filter designer to evaluate different models and inputs to the filter in order to answer more general questions about model order and parameterization for a particular problem.

References

[1]Gelb, A., *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974, pp. 255–260.

[2]Maybeck, P. S., *Stochastic Models, Estimation, and Control*, Vol. 1, Addison Wesley Longman, Reading, MA, 1989, p. 337.

[3]Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes*, Vol. 1, Cambridge Univ. Press, Cambridge, England, U.K., 1989.

[4]Ljung, L., *System Identification Theory for the User*, Prentice–Hall, Upper Saddle River, NJ, 1987, pp. 471, 472.

[5]Oshman, Y., and Shaviv, I., "Optimal Tuning of a Kalman Filter Using Genetic Algorithms," AIAA Paper 2000-4558, Aug. 2000.

[6]Nelder, J. A., and Mead, R., "A Simplex Method for Function Minimization," *Computer Journal*, Vol. 7, No. 4, 1965, pp. 308-313.

[7]Powell, T. D., Feess, W. A., and Menn, M. D., "Evaluation of GPS Architecture for High Altitude Spaceborne Users," *Proceedings of the Institute of Navigation 54th Annual Meeting*, Inst. of Navigation, Alexandria, VA, 1998, pp. 157–165.