Numerical Approximations and Other Structural Issues in Practical Implementations of Kalman Filtering

Thomas H. Kerr

Abstract. Getting incorrect results at the output of a Kalman filter (KF) simulation or hardware implementation can be blamed on (1) use of faulty approximations in the implementation, or (2) on faulty coding/computer programming or (3) may actually be due to theoretical details of what should be implemented in the application being incorrectly specified by the analyst (especially since some errors still persist and propagate in the published literature that the analyst may have used as a starting point). Handling situations (1) and (3) will first be discussed here. Although situation (2) is initially impossible to distinguish from the effects of (1) and (3) for a new candidate KF software implementation, any residual problems present can be ferreted out by first eliminating (1) and (3) as possibilities for contamination and problems falling under situation (2) may be further isolated (for remedy) by using certain test problems of known analytic closed-form solution for software calibration/check-out in the manner discussed as my original unique approach to (IV&V) Independent Verification and Validation (completely compatible with DOD-STD-SDD/2167/2168A/973/499B/490B methodology) for Kalman filter code. The techniques espoused here are universal and independent of the constructs of particular computer languages and were honed from years of experience in cross-checking Kalman filter implementations (both my own and those of others) in several diverse commercial and military applications (and implementation languages).

§1 Introduction

Over the past thirty years, Kalman filters (KF) have been used in telephone line echo-cancelers, missiles, aircraft, ships, submarines, tanks that shoot-on-the-run, air traffic control (ATC) radars, defense and targeting radars, Global Position System (GPS) sets, and other standard navigation equipment. In recent years, GPS/Kalman filter combinations in conjunction with laser disk-based digital map technology is being considered for use in future automobiles (as well as in ships

Approximate Kalman Filtering
Guanrong Chen (ed.), pp. 193–220.

ISBN 981-02-1359-X

using displays rather than paper charts) to tell the driver/pilot where he is and how to get where he wants to be. Commercial products as well as military vehicles and platforms rely on Kalman filters. Computers are used to implement Kalman filters and to test out their performance (assess the speed of response and the accuracy of their estimates) beforehand in extensive simulations. There is evidently considerable commercial value in understanding Kalman filters both as a developer and as a potential user [30].

I seek to weave the thread of the story here but, due to space limitations, will defer to my references for more elaboration (which provide pointers to the further contributing precedents of others, thus serving as analytical stepping stones in the evolution). I will be uncharacteristically terse when discussing a topic here that I have previously published and already extensively discussed elsewhere in the open literature. The techniques espoused here were honed from years of experience in cross-checking Kalman filter implementations (both my own and those of others) in several diverse commercial and military applications from first hand knowledge, having worked directly with C-3 Poseidon submarines' Ships Inertial Navigation System (SINS) 7 state Con-B STAtistical Reset (STAR) filter, C-4 Trident submarines' 14 state Electro-magnetically Supported Gyro Monitor (ESGM) Reset filter and 15 state SINS Correction filter, earlier vintage minesweeper 19 state PINS filter, 13 state Passive Tracking Algorithm (PTA) filter for sonobuoy target tracking, 15 and 18 state Singer-Kearfott and Hughes candidate Class B JTIDS filters (filter parameters such as INS gyro drift-rates, biases, and scale-factor errors are classified for military applications, otherwise standoff targeting and bombing accuracy and radio-silent rendezvous capability could be inferred; however, such gyro and accelerometer parameter information should be reported for clarity in civilian applications according to new specification standards currently being revised by the IEEE AES Gyro and Accelerometer Panel), 12 state filter for Electronic Terrain Board analysis, 22 state Multi-Band multi-Frequency Airborne Radio System (MF-BARS) filter predecessor to ICNIA for the Advanced Tactical Fighter, various GPS filters [12, Table III], angle-only tracking filters and other 6 state exoatmospheric and 7 state endoatmospheric Reentry Vehicle (RV) tracking filters for radar, etc.

§2 Some numerical approximation issues that arise in Kalman filtering

A Kalman filter (see Figure 1) is an efficient and convenient computational scheme for providing the optimal estimate of the system state and an associated measure of the goodness of that estimate (the variance or covariance). In order to implement a KF, the actual continuous-time system must be adequately characterized by a linear (or linearized) ordinary differential equation model, represented in state space at time t in terms of a vector $\mathbf{x}(t)$, and having associated initial conditions specified, and availing sensor output measurements $\mathbf{v}(t)$ (functions of the state plus additive measurement noise). It is mandatory that the KF itself actually contain within it an analytical mathematical model of the system and sensors in order to

perform its computations (designated as a model-based estimator), and it must possess a statistical characterization of the covariance intensity level of the additive white Gaussian measurement and process noises present as well to enable an implementation. Here, we should remark that the Central Limit Theorem is usually invoked from statistics [25, pp.238-240] to justify that a number of contributing minor effects can frequently sum up to a net effect that is Gaussian in distribution, even when the constituent components are not (i.i.d.) independent and identically distributed. Care is sometimes needed to be aware of when a necessary condition on the 3rd moments of the random variables contributing to the sum is in danger of being violated [22, pp.66-73] (as occurs with the bell-shaped Cauchy distribution) otherwise Gausianess is never attained and other approaches need to be used.

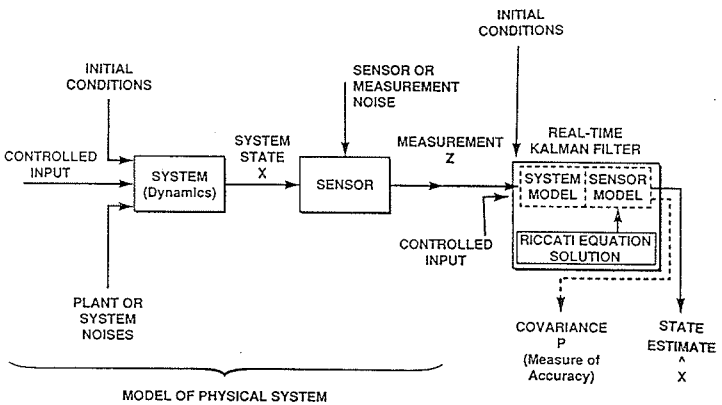

Figure 1. Overview functional block diagram of the internal structure of a Kalman filter.

A simplified overview of the principles of operation of a Kalman filter has been treated in [17, Sec.V, pp.943-944], [20, Sec.IA] and, from my perspective, is what constitutes the essence of a Kalman filter mechanization. References [3,9,5] all address important numerical approximation issues that *sometimes* arise in Kalman

filtering such as numerical sensitivities and ill-conditioning, adverse effects due to use of quantized data, the nature of the matrix inversion algorithm or approximation utilized within a real-time KF mechanization and its subsequent effect on convergence, respectively, each of which can corrupt KF performance and degrade its tracking accuracy in some instances. However, the approach taken here will be to instead consider the more prevalent 1st order issues usually encountered in a reasonable (but typically clumsy) software implementation attempt and leave pointers to references to indicate where more detailed information may be found on the more sophisticated topics that arise less frequently but are important never-the-less.

2.1 Typical errors and/or bad approximations occurring in Kalman filter code of otherwise good quality

Listed below are several prevalent departures from the ideal involving use of expedient approximations and simplifying assumptions that one must be alert to avoid lest they taint or corrupt KF output results. Consider the following possible shortcoming's in KF code (each having been previously observed in both government (DoD) and commercial KF packages and code implementations):

1. Some KF software/covariance analysis implementations don't use the *exact* discrete-time equivalent to continuous-time white Gaussian system noise (plant or process noise) $\xi(t)$ [24, p. 171, Eq. (4-127b)] represented by:

$$Q_k = \int_{t_k}^{t_{k+1}} \Phi(t_{k+1},\tau) Q_c(\tau) \Phi^\top(t_{k+1},\tau)\, d\tau, \tag{1}$$

(where $\Delta = t_{k+1} - t_k$ and $\Phi(t,\tau)$ is the associated system transition matrix) as an operation on the continuous-time white process noise covariance intensity matrix, Q (or Q_c), as in [20, Eq. (5)], where equation (1) simplifies for time-invariant systems (obtained by the steps depicted in [17, Sec.II]) to be:

$$Q_d = e^{A\Delta} \left[\int_0^\Delta e^{-A\tau}\, B\, Q\, B^\top\, e^{-A^\top \tau}\, d\tau \right] e^{A^\top \Delta}\ \delta_{kj}\ , \tag{2}$$

where the above Kronecker delta is defined as

$$\delta_{kj} = \begin{cases} 1, & \text{if } k = j \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

An offending software code implementation (of the type being cautioned against) instead uses the following Kalman approximation:

$$Q'_d = \Delta\, Q\ , \tag{4}$$

which is an *uncalibrated* approximation [17,27] and, for implementation, Q'_d or Q_d is usually further factored (via a Choleski decomposition if necessary) as $Q_d = BB^\top$, where here B is used as the process noise gain matrix, according to the convention of [29] while the underlying white noise originally simulated is of unit variance. It is demonstrated in [17, following Eq. (40)] (also see Nov.'91 update) just how bad the effect of this approximation of equation (4) can be by the degree of error incurred using Q'_d as compared to Q_d via equations (1) or (2). However, the approximation of Eq. 4 may still be satisfactory for some special application situations.

2. The act of taking the time step to be constant in any KF mechanization corresponds to external position fixes being obtained periodically (while actual KF theory is more flexible than this and real world practicalities don't always strictly adhere to this periodic structure) so implementers frequently compensate by resorting to data time-tags and appropriate extrapolation to the desired time or by measurement data averaging, explained in [8, p. 291];
3. The transition matrix calculation for converting the continuous-time n-state model description to discrete-time, historically adaptively tailors the number of terms retained in the Taylor series by using either too coarse a norm (see [29]) or an invalid norm [17, pp.938-939]. A tighter bound for this purpose has been derived from considerations of both column-sum and row-sum norms in [11] and, additionally, it is prudent to also set an upper limit on the total number of terms from the Taylor series expansion allowed to be used in calculating the transition matrix so that the computation can't run away (otherwise it could incur numerous *overflow*'s due to the effect of accumulated roundoff).
4. The *transition matrix* used throughout the computer run is frequently calculated only once (such as in [29]), up front as a pre-processing step, then retained as being constant (while a variation more appropriate for many applications but not possible with a simplistic software implementation is to relinearize $\mathbf{a}(\mathbf{x})$ (occurring within the ordinary differential equation $\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}, \underline{\xi}, t)$ describing the system) at each new time point as $A(t_k)$ and either recalculate the matrix exponential to provide the new *transition matrix* at each relinearization or else just use the first few Taylor series terms of the matrix exponential as $I + A(t_k)(t_{k+1} - t_k)$ to approximate the *transition matrix* at this new time point, an especially prevalent solution found in many real-time applications).
5. Fallacious versions of tests of matrix positive definiteness/semi-definiteness are prevalent in Kalman filter code and in target tracker code. Two faulty algorithms that have been encountered in actual use as reasonableness tests for covariance matrices are explicitly identified and warned against in [19, Secs. III and IV], with appropriate theoretical fixes suggested (based on use of SVD variants or Choleski factorization).

Finally, alternative approaches that have evolved to justify the technique of equation (4) as the appropriate *approximate* discrete-time process or plant noise

covariance intensity matrix Q'_d to be used for computer simulations is traced in the Nov.'91 update to [17]. To clarify for actual applications, the usual rule-of-thumb (recommended by Industry) to use for discrete-time white noise simulation, is that the continuous time white process noise $Q_{continuous}$ has units of "$\frac{\text{(numerator units)}^2}{\text{time units}}$" so one needs to multiply throughout by Δ (with time units) that cancel the time units in the denominator units of $Q_{continuous}$ to yield Q'_d, with exclusively (numerator units)^2" for discrete-time sample-data white process noise. (Note that in equation (1), the exact expression for Q_d rids itself of time units in the denominator via the indicated integration with respect to time.) This final value is then used in actually performing a discrete-time simulation performance evaluation on a computer as the appropriate approximate technique (verified here by a units check and by allowing a type of correspondence for the white noise $Q_{continuous}$, also associated with a Dirac delta function, and the white noise Q'_d, associated with the more benign, less pathological, Kronecker delta function (that unlike the Dirac delta function doesn't blow up or need special interpretation for rigor as a *functional of bounded convex support* using Schwartz's *Theory of Distributions* (1947)).

2.2 An approach for debugging linear Kalman filter software

The importance of this section is that subsequent software verifiers, when faced with validating newly coded or newly procured Monte-Carlo simulator subroutine software modules and Kalman filters of their own, can treat the entire exercise as one of confirming the proper performance behavior of the new modules merely as an exercise with black boxes. Time can then be saved by just confirming the outputs corresponding to the designated low-dimensional test cases of known closed-form solution provided herein and matching critical intermediate computational benchmarks (without having to necessarily further probe the internal theoretical intricacies that are already justified here, in [17] and in [15], where the veracity and utility of these test cases is established and explained in more detail) but can instead check the code, with helpful clues as to the real software culprits and bugs being revealed by these recommended tests when output results don't jibe. Thus, the software verification/validation job is simplified by using the results presented here and used to pinpoint or isolate any problems that exist in the code. This entire exercise of using simple transparent test problems may be interpreted as an initial calibration of the available software before proceeding to use the parameters of the actual application.

However, before the KF code can be validated as performing properly, or in case of known errors, before the source can be pin-pointed, first the inputs to the KF must be validated as being exactly what was intended. To this end, we first turn our attention to validating the Monte-Carlo simulator, as addressed next.

2.2.1 Capabilities designed into the simulator

A state-variable based Monte-Carlo simulator, of the form depicted in Figure 2, was developed to support AR process emulation for testing the performance of multi-channel Linear Prediction algorithms (of the Maximum Entropy type) for spectral estimation and also for testing the adequacy of KF trackers. This simulator possesses the following *modern* features:

- Incorporates "*exact* discrete-time equivalent of continuous-time white noise";
- Offers option of using the more efficient direct calculation of steady-state initial conditions corresponding to stationary behavior of the underlying random process (without having to iterate to steady-state to avoid the nonstationary initial transient);
- Offers option of having additive (stationary white Gaussian) output measurement noise present (thus creating a type of ARMA process);

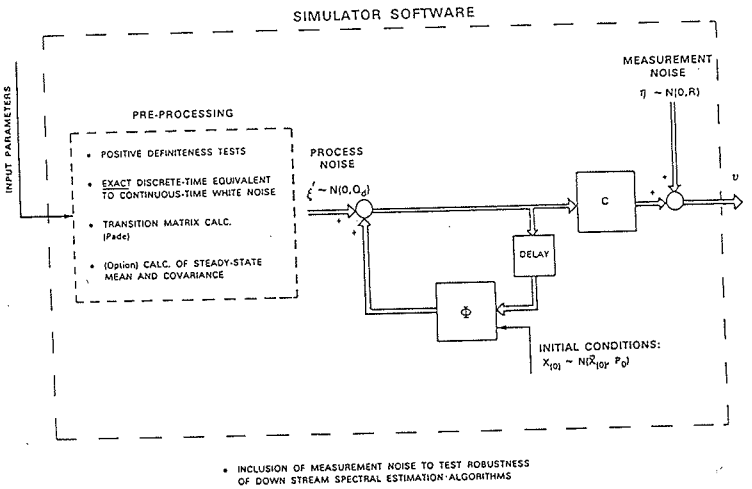

Figure 2. State-variable Markov-based Monte-Carlo simulator.

- Isn't restricted to use of only diagonal covariances for noises or for initial condition covariances;
- If covariances are not diagonal, the program internally automatically checks to verify that the covariances possess the requisite "positive definiteness" property (via use of the Singular Value Decomposition (SVD) in the manner indicated in [19; 13 (p. 504); 18; 14 (Sec.III, p. 63)]), otherwise diagonal covariances are merely verified not to have zeroes or negative numbers on the principal diagonal;
- Calculates transition matrix by more accurate Pade approximation (offering two validated options along these lines [17, Sec.III]) rather than through use of a Taylor series expansion for this purpose [17, Fig.1];
- Can handle nonzero means for both noises and initial conditions;
- Outputs final pseudo-random noise (PRN) generator seed value to enable continuity of use via allowable dovetailing of output sample functions if further prolonged sample function history is subsequently pursued (which uses this PRN seed during subsequent start-up).

2.2.2 Verifying the simulator proper

The overall structure of the simulator is depicted in Fig. 2. Using the input parameters of Test Case 1, as depicted in Table 1, the intermediate outputs provided by the software implementation were verified to be correct. The specific features of the software implementation that were confirmed using Test Case 1 are detailed in the second column from the left in Table 2. The importance of using Test Case 1 and the aspects that it reveals are described next.

Certain matrices known as "idempotent" matrices have the unusual property that when multiplied times itself again yields itself as the result:

$$A\,A = A. \tag{5}$$

The non-trivial system matrix of Test Case 1 exhibits this property. The present application in software verification is a neat application of idempotent matrices being used to construct test matrices for verifying the transition matrix algorithmic implementations that are used for computer computation of e^{Ft}. The utility of these test matrices is that the resulting analytically derived expression for e^{Ft} is conveniently in closed-form for $F = A$. Hence the performance of a general e^{Ft} subroutine implementation can ultimately be gauged by how close it comes to achieving the known ideal exact solution.

Using the representation of a matrix exponential, defined in terms of its Taylor series , but evaluated with an idempotent matrix A having the property of equation (5) being substituted along with time-step Δ; the matrix Taylor series expansion of $e^{A\Delta}$ now yields

Table 1. Summary of parameters of test case models used in validation tests of primary software modules.

Case No.	Test Case 1	Test Case 2	Test Case 3	Test Case 4
Step Size DEL (Δ)	0.405	0.5	0.5	1
System Matrix A	$\begin{bmatrix} 1/3 & -1/3 & 1/3 \\ -1/3 & 1/3 & -1/3 \\ 1/3 & -1/3 & 1/3 \end{bmatrix}$	$\begin{bmatrix} -5 & -1 \\ 6 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	—
Transition Matrix $e^{F\Delta}$	$\begin{bmatrix} 1.166 & -0.166 & 0.166 \\ -0.166 & 1.166 & -0.166 \\ 0.166 & -0.166 & 1.166 \end{bmatrix}$ as calculated	$\begin{bmatrix} -0.0664 & -0.1447 \\ 0.8685 & 0.6574 \end{bmatrix}$ as calculated	$\begin{bmatrix} 1 & 0.5 \\ 0 & 1 \end{bmatrix}$ as calculated	$\begin{bmatrix} 0.34 & -0.22 & -0.75 \\ 0.65 & & 0.55 \end{bmatrix}$ as entered
NDIM	NDIM = 3	NDIM = 2	NDIM = 2	NDIM = 2
MDIM	MDIM = 2	MDIM = 2	MDIM = 2	MDIM = 2
Process Noise Covariance Intensity Matrix Q	continuous time version $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	continuous time version $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	continuous time version $\begin{bmatrix} 10^{-8} & 0 \\ 0 & 10^{-8} \end{bmatrix}$	Discrete time version $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Observation Matrix C	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Measurement Noise Covariance Intensity Matrix R	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 10^{-8} & 0 \\ 0 & 10^{-8} \end{bmatrix}$	$\begin{bmatrix} 10^{-8} & 0 \\ 0 & 10^{-8} \end{bmatrix}$
Initial mean X_0	$\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$	$\begin{bmatrix} 10 & 4 \end{bmatrix}^T$	$\begin{bmatrix} 0 & 0 \end{bmatrix}^T$
Initial Covariance P_0	$\begin{bmatrix} 6 & 2 & 1 \\ 2 & 8 & 3 \\ 1 & 3 & 12 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 10^{-8} & 0 \\ 0 & 10^{-8} \end{bmatrix}$	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$$
\begin{aligned}
e^{A\Delta} &= \sum_{k=0}^{\infty} \frac{A^k}{k!}\Delta^k \\
&= I + \frac{A}{1!}\Delta + \frac{A^2}{2!}\Delta^2 + \frac{A^3}{3!}\Delta^3 + \cdots \\
&= I + A(\frac{\Delta}{1!} + \frac{\Delta^2}{2!} + \frac{\Delta^3}{3!} + \cdots) \\
&= I + A(1 + \frac{\Delta}{1!} + \frac{\Delta^2}{2!} + \frac{\Delta^3}{3!} + \cdots - 1) \\
&= I + A(e^{\Delta} - 1)\,, \qquad (6)
\end{aligned}
$$

as explained in [17, Sec.IV]. Thus, the closed-form exact expression for the transition matrix corresponding to idempotent system matrices is as depicted in the last line of equation (6) as a finite two step operation involving just a scalar multiplication of a matrix and a single matrix addition (as compared to an infinite series that must be truncated in the case of standard software implementations for the case of more general matrices).

Using the result of equation (6) for idempotent matrices within the more general expression of equation (2), allows this expression for the required discrete-time process noise covariance to be evaluated analytically in closed-form as:

$$
\begin{aligned}
Q_d &= \left[I + A(e^{\Delta} - 1)\right] \int_0^{\Delta} \left[I + A(e^{-\tau} - 1)\right] BQB^{\top} \\
&\quad \times \left[I + A^{\top}(e^{-\tau} - 1)\right] d\tau \left[I + A^{\top}(e^{\Delta} - 1)\right] \\
&= \left[I + A(e^{\Delta} - 1)\right] \int_0^{\Delta} \left[BQB^{\top} + (ABQB^{\top} + BQB^{\top}A^{\top})(e^{-\tau} - 1)\right. \\
&\quad \left. + ABQB^{\top}A^{\top}(e^{-2\tau} - 2e^{-\tau} + 1)\right] d\tau \left[I + A^{\top}(e^{\Delta} - 1)\right] \\
&= \left[I + A(e^{\Delta} - 1)\right] \left[BQB^{\top}\Delta + (ABQB^{\top} + BQB^{\top}A^{\top})(1 - e^{-\Delta} - \Delta)\right. \\
&\quad \left. + ABQB^{\top}A^{\top}\left(-\frac{3}{2} - \frac{1}{2}e^{-2\Delta} + 2e^{-\Delta} + \Delta\right)\right] \left[I + A^{\top}(e^{\Delta} - 1)\right]\,. \qquad (7)
\end{aligned}
$$

This is a new result that is also useful as a confirming check for software implementations of equation (2). Here, we remark that along a different line, something similar to equation (2) can be computed for numerically evaluating Q_d for *any* constant matrix A, not just for idempotent matrices, by (1) expanding $e^{-A\tau}$ into its matrix Taylor series, (2) by performing the indicated multiplications of the two series within the integrand, (3) by subsequently performing term-by-term integration, and then (4) by retaining enough terms of the final series to be used to provide sufficient accuracy in actual numerical calculations.

Using the parameters of Test Case 2, as depicted in Table 1. The specific features of a software implementation that can be confirmed using Test Case 2 are detailed in the third column from the left in Table 2. Test Case 2 has an easy to determine closed-form expression for the transition matrix, for Q_d, for the steady-state Lyapunov equation, and for the ideal output power spectrum [15, Appendix A].

Using the parameters of Test Case 3, as depicted in Table 1 (with closed-form expressions for the solution being a straight line with slope 4 and intercept 10 and with such inconsequentially low magnitudes of the noises), the intermediate outputs provided by a software implementation can be verified to be correct. The specific features of the software implementation that can be confirmed using Test Case 3 are detailed in the fourth column from the left in Table 2. Actual extremely regular essentially *deterministic* sample functions obtained for the underlying known unstable system can conveniently be used to check at a high level that the output is exactly correct. Besides confirming the outputs of the simulator with an easily recognizable expected answer (as contrasted to Test Cases 1, 2, and 4, which provide random noise corrupted sample functions that can be confirmed at the aggregate level only from statistical properties that are a byproduct of downstream KF tracking or spectral estimation), this Test Case 3 also allows a programmer to calibrate (and correct) their plot routines and his scale conversion for output plots, if necessary.

Using the parameters of Test Case 4, as depicted in Table 1, the intermediate outputs provided by a software implementation can be verified to be correct. The specific features of a software implementation that can be confirmed using Test Case 4 are detailed in the fifth column from the left in Table 2. The main purpose of this last test case is to be able to handle the situation of providing prescribed multi-input/multi-output (MIMO) complex random process output with specified cross-correlation between output channels. This was needed in [15] in verifying the performance of alternative Maximum Entropy spectral estimators down-stream of the simulator (operating on its outputs), which, like a KF, deal only with first and second order statistics. The correct answer for 2-channel spectral estimation should appear as in Figure 3. Certain modern tracking radars use coherent phase processing, also known as coherent integration (where both magnitude and phase are accounted for in the summation of signal returns but where the distinction arises of having to keep track of real and imaginary components, instead of merely needing to keep track of magnitude alone, as conventional radars do), which jointly treats Primary Polarization (PP) returns in conjunction with Orthogonal Polarization (OP) returns and utilizes the additional target information provided from the cross-correlation of these two separate channels.

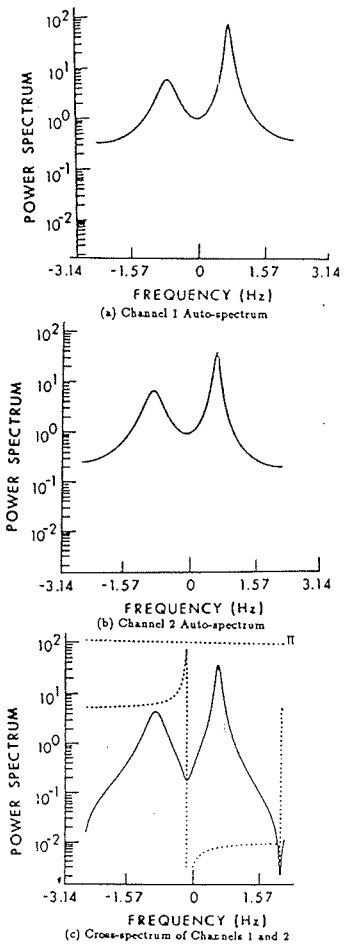

Figure 3. True spectrum for the two channel complex Case 4.

Restating for emphasis, the modern simulator design, discussed in this section, was pursued so that only a fairly exact mechanization would be used so that the input to the KF is precisely known. This was sought as a reliable testbed that avoids use of uncalibrated approximations in order to avoid confusing artifacts of simulator approximations with possible cross-channel feed through (that multichannel spectral estimation implementations are also known to frequently exhibit as a weakness or vulnerability) and which can adversely affect KF testing as well for the same reasons of uncalibrated cross-correlations being present.

2.2.3 Confirmation of software structural validity using augmented test cases of known closed-form solution

A difficulty, as discussed in [16, Sec.I], is that most closed-form KF covariance solutions are of either dimension 1 or 2 (as in [8, pp.138-142, pp.243-244, p.246, pp.255-257, pp.318-320]) or 3 (as in [26]). To circumvent this dimensional mismatch to higher dimensional real applications that may be hard-wired, we can achieve the dimension n goal by augmenting matrices and vectors with a concatenation of several existing test problems. Use of only totally diagonal decoupled test problems is notorious for being too benign or lenient and not taxing enough to uncover software implementation defects (when the problems exist in the portion of the code that handles cross-term effects). Augmenting either several low-dimensional 2-state problems or fewer 3-state problems is the way to proceed in order to easily obtain a general n-state non-trivial non-diagonal test problem. A confirmation that this proposed augmentation is valid in general is provided next for a closed-form steady-state radar target tracking solution that was successfully used as a check on the software implementation of [29].

An initial worry in adjoining the same 3-state problem with itself relates to whether "controllability and observability" are destroyed, while the 3-state problem by itself does possess the requisite "controllability and observability." "Controllability and observability" conditions, or at least more relaxed but similar "stabilizability and detectability" conditions [21, pp.62-64, pp.76-78, pp.462-465], need to be satisfied in order that the covariance of a KF be well-behaved [8 (p.70, p.142), 21]. The following mathematical manipulations establish that such an adjoining of two 3-state test problems does not destroy the "controllability and observability" of the resulting 6-state test problem even though it already exists for the 3-state test problem by itself.

First consider the 3-state test problem of [26] of the following form:

$$\mathbf{x}_{(3\times 1)} = \begin{bmatrix} \text{position} \\ \text{velocity} \\ \text{acceleration} \end{bmatrix}, \tag{8}$$

with

$$\begin{aligned} \dot{\mathbf{x}} &= A_1\mathbf{x} + B_1\underline{\xi}, \quad \underline{\xi} \sim \mathcal{N}(0, Q_1), \\ \mathbf{v} &= C_1\mathbf{x} + \underline{\eta}, \quad \underline{\eta} \sim \mathcal{N}(0, R_1), \end{aligned}$$

and assumed to be already satisfying Kalman's "controllability and observability" rank test criteria [8, p.70], respectively, as

$$\text{rank}\,[B_1 \vdots A_1B_1 \vdots A_1^2B_1] = n_1 = 3\,, \tag{9}$$

$$\text{rank}\,[C_1^{\mathsf{T}} \vdots A_1^{\mathsf{T}}C_1^{\mathsf{T}} \vdots [A_1^{\mathsf{T}}]^2C_1^{\mathsf{T}}] = n_1 = 3\,. \tag{10}$$

Now the augmented system of the form

$$\mathbf{x} = \begin{bmatrix} \text{position} \\ \text{velocity} \\ \text{acceleration} \\ \cdots \\ \text{position} \\ \text{velocity} \\ \text{acceleration} \end{bmatrix}, \tag{11}$$

with

$$\dot{\mathbf{x}} = \begin{bmatrix} A_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & A_1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} B_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & B_1 \end{bmatrix} \begin{bmatrix} \underline{\xi} \\ \cdots \\ \underline{\xi} \end{bmatrix}, \tag{12}$$

$$\mathbf{v} = \begin{bmatrix} C_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & C_1 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & 1 \end{bmatrix} \begin{bmatrix} \underline{\eta} \\ \cdots \\ \underline{\eta} \end{bmatrix} \tag{13}$$

has system, process noise gain, and observation matrices, respectively, of the form

$$A_2 = \begin{bmatrix} A_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & A_1 \end{bmatrix}, \tag{14}$$

$$B_2 = \begin{bmatrix} B_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & B_1 \end{bmatrix}, \tag{15}$$

$$C_2 = \begin{bmatrix} C_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & C_1 \end{bmatrix}. \tag{16}$$

In testing for controllability of this augmented system, form

$$\begin{aligned} &\text{rank}\begin{bmatrix} B_2 & \vdots & A_2B_2 & \vdots & A_2^2B_2 & \vdots & A_2^3B_2 & \vdots & A_2^4B_2 & \vdots & A_2^5B_2 \end{bmatrix} = \\ &\text{rank}\begin{bmatrix} B_1 & \vdots & 0 & \vdots & A_1B_1 & \vdots & 0 & \vdots & A_1^2B_1 & \vdots & 0 & \vdots & \text{other} \\ \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots \\ 0 & \vdots & B_1 & \vdots & 0 & \vdots & A_1B_1 & \vdots & 0 & \vdots & A_1^2B_1 & \vdots & \text{stuff} \end{bmatrix} = \\ &\text{rank}\begin{bmatrix} B_1 & \vdots & A_1B_1 & \vdots & A_1^2B_1 & \vdots & 0 & \vdots & 0 & \vdots & 0 & \vdots & \text{other} \\ \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot & \cdots \\ 0 & \vdots & 0 & \vdots & 0 & \vdots & B_1 & \vdots & A_1B_1 & \vdots & A_1^2B_1 & \vdots & \text{stuff} \end{bmatrix} \\ &= 3+3=6\,. \end{aligned} \tag{17}$$

In the next to the last line of equation (17), the columns of the Controllability Grammian are rearranged for convenience to provide the necessary insight. Permuting columns of a matrix doesn't alter its rank but can alter at-a-glance conclusions. Since we are able to show that the augmented system rank is 6, this system is confirmed to be controllable. A similar conclusion (on the requisite observability being satisfied) can be obtained by identical steps using the duality that exists between controllability and observability results and the associated forms of arguments or proofs when similar matrix structures, such as are present here, are involved. The above described augmented system of equations (12) and (13) can be used with

$$R_2 = \begin{bmatrix} R_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & R_1 \end{bmatrix}, \tag{18}$$

$$Q_2 = \begin{bmatrix} Q_1 & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & Q_1 \end{bmatrix}, \tag{19}$$

$$P_2(0) = \begin{bmatrix} P_1(0) & \vdots & 0 \\ \cdots & \cdots & \cdots \\ 0 & \vdots & P_1(0) \end{bmatrix}, \tag{20}$$

since now the augmented system has been demonstrated above to be both "observable and controllable" and the measurement noise covariance R_2 of equation (18) to be utilized is positive definite. This final observation allows us to use this 6-state augmented test problem with confidence to check out the software implementation as it is currently configured without making any further changes to the software.

2.2.4 Summary of test coverage analytically provided here

An overview of the complete software test coverage offered here through selective use of analytic closed-form "Test Cases of known solution" is provided in Table 2. The utility of this coverage was discussed in Sec. 2.2.2. All items indicated in Table 2 must be successfully validated.

By using these or similar examples, certain qualitative and quantitative aspects of the software implementation can be checked for conformance to anticipated behavior as an intermediate benchmark, prior to modular replacement with the various higher-order matrices appropriate to the particular application. This procedure is less expensive in CPU time expenditure during the software debug and checkout phase than using the generally higher n-dimensional matrices of the intended application since the computational burden is generally at least a cubic polynomial in n during the required solution of a Matrix Riccati equation for the associated covariances (also needed to specify the Kalman gain at each time-step). The main contribution of these Test Cases is that one now knows what the answers should be beforehand and is alarmed if resolution is not immediately forthcoming from the software under test. *Warning*: correct answers could be "hardwired" within candidate software under test, but appropriate scaling of the original test problems to be used as inputs can foil this possible stratagem of such an unscrupulous supplier/developer.

The benefits of using these recommended or similarly justified test cases are the reduced computational expense incurred during software debug by using such low-dimensional test cases and the insight gained into software performance as gauged against test problems of known solution behavior. However, a modular software design has to be adopted in order to accommodate this approach, so that upon completion of successful verification of the objective computer program implementation with these low-dimensional test problems, the matrices corresponding to the actual application can be conveniently inserted as replacements without perturbing the basic software structure and interactions between subroutines. Even time-critical, real-time applications can be validated in this manner even when using matrix dimensions that are "hardwired" to the particular application by tailoring to the specified dimension using the technique of Section 2.2.3.

2.2.5 Specifying a Kalman filter covariance test problem

The particular parameter values to be used for F_1, B_1, and C_1 in equations (12) and (13) (as laid out following equations (2) and (3) of [26]) are

$$\Phi_1 = \begin{bmatrix} 1 & T & T^2/2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \tag{21}$$

(corresponding to $A_1 = I_3$, where $\Phi_1 = e^{A_1 T}$),

Table 2. Simulator testability coverage matrix.

FUNCTION	CASE 1	CASE 2	CASE 3	CASE 4
Transition Matrix Computation: Pade (Ward's Algorithm)	√	√		
Transition Matrix Computation: Pade (Kleinman's Algorithm)	√	√		
Q_d Computation: Discrete-Time Equivalent of Continuous-Time White Noise	√	√		
Steady-State Computation of Initial Condition Mean		√		
Steady-State Computation of Initial Condition Covariance (Lyapunov Equation Solution)		√		
Verification of SVD-based Positive Definiteness Test for Nondiagonal Matrices	√			
Verification of Abbreviated Positive Definiteness Test for Diagonal Matrices	√	√	√	√
Checked Process Noise Calculations as Output from Random Number Generator	√	√		
Checked Measurement Noise Calculations as Output from Random Number Generator	√	√		
Checked Recursive Calculation of all Constituent Components of Entire Random Process Over Several Iterations	√	√		
Checked Proper Handling of PRN Seed	√	√		
Verification of Stable Sample Functions Indicative of Stationary Process		√		√
Verification of Unstable Sample Functions Indicative of Nonstationary Process	√		√	
Obvious Aggregate High Level At-A-Glance Confirmation From Output that all Functions Work Properly in Concert			√	
Confirmation of Identical Results When Complex Version of Software Enabled	√	√	√	
Eventual Confirmation of Proper Sample Function Statistics from Downstream Spectral Estimation Software Module Outputs		√		√

$$B_1 = \begin{bmatrix} T^3/6 \\ T^2/2 \\ T \end{bmatrix}, \tag{22}$$

and

$$C_1 = [\,1\ 0\ 0\,], \tag{23}$$

with

$$Q_1 = \sigma_a^2, \tag{24}$$

and

$$R_1 = \sigma_x^2, \tag{25}$$

where, in the above, T is the fixed time-step and σ_a and σ_x are parametrization conventions used in [26]. Tractably determining the particular values of T, σ_a, and σ_x to be used here is the contribution of the Appendix section that provides the steady-state covariance via the parameterized methodology of [26]. The computed steady-state KF covariance immediately before and after a noisy position measurement update are, respectively, as provided in equation (66) (res. 67) and (74) (res. 75).

Notice where the KF residuals occur in [17, Fig.4]. In verifying and debugging an actual KF software implementation, these residuals are monitored and used as a gauge-of-goodness and indicate good tracking performance when they become "small." The idea being that the measurements $\mathbf{v}_k$ match the model representation $C_k\hat{\mathbf{x}}_{k|k-1}$ fairly closely when the residuals are "small." However, since residuals are never identically zero, the question is "how small is small enough?" (see [1] for an appealing explicit statistical test on the residuals using Chi-square statistics with appropriately specified degrees-of-freedom for an assortment of likely test conditions that can occur.) Residuals (sometimes called *innovations*) will almost always initially decrease as the initial transient settles out. "Small residuals" are necessary but not sufficient indicators of good KF performance and similar statements can be made for having statistically white residuals (for instance, see [4,23] which offer an example of a KF exhibiting white residuals despite known use of an incorrect system model but which also incurs an anomalous bias as the clue that something is wrong). When possible, as with simulations, one should juxtapose the time evolution of any critical system states along side their KF estimates to see how closely the estimates are following the actual quantities of interest as a more encompassing gauge of proper KF performance. The only problem sometimes encountered in certain sensitive applications is that actual estimates may be classified while residuals may be unclassified, in which case attention centers on the residuals in unclassified presentations as a default in justifying good filter performance. For actual real system data, the true system state uncontaminated by measurement noise is seldom available for confirming comparisons of proximity so use of residuals must suffice in this situation also.

Besides being used numerous times in the past to validate DoD KF software implementations, the techniques espoused in Sec. 2.2 were used to test the commercial KF code of [29] for the PC. This commercial package initially satisfied all test cases invoked (Test Case 4 was skipped, as not being applicable, since this code wasn't designed to handle the situation involving complex number computations). Proceeding to use the augmentation of Section 2.2.3, this commercial KF bombed when attempting to run applications with state sizes greater than 10. Source of the problem was traced to a software bug that was revealed to be in the Monte-Carlo simulator *AVBSIM.BAS*, as one of the 13 modular subroutines. Proper declaration of index on initial condition $\bar{\mathbf{x}}_0$ had been overlooked by the supplier so the software (in BASIC) had defaulted to 10. Note that by later using different designators to correct another oversight, this initial condition, as used in initializing the simulator, had to be distinguished from its use in initializing the filter proper *AVBFILTR.BAS*. This oversight was easily corrected in the source code and it then successfully handled system models larger than 10. The original code was also enhanced to (1) read P_0 from a file, instead of requiring excruciating hand entry from a keyboard at run time, (2) output results to a file so a more capable plot package, such as $EASYPLOT^{TM}$, could be used for final display.

2.3 An approach for debugging nonlinear filter software implementation

An exact finite-dimensional optimal nonlinear filtering test case of the type discovered by Benes [2] and extended by Daum [6] (with a recent rigorous update in [28]) may suffice for IV&V in the same manner as Section 2.2 by providing collaborative comparison of outputs to verify performance of a general EKF implementation (instantiated with the same test case) if both implementations agree (sufficiently) for this simple test. This proposed manner of use for EKF software verification would be in keeping with the overall software test philosophy being espoused here. We remark that another nonlinear filtering example with a finite-dimensional implementation, not covered within the situations addressed in [2] and [6], is for scalar system $\dot{x} = f(x) + g\,\xi(t)$, where $E\,[\xi(t)\xi(s)] = q\,\delta(t-s)$ and $f(x) = -\frac{x}{1+x^2}$. The verifiable asymptotic solution in the limit as t goes to infinity of the associated Fokker-Planck or forward Chapman-Kolmogorov equation (defined in [10, pp.126-130]) is $p(x,t|z,s) = \frac{c}{(1+x^2)^{c_2}}$, where $c_2 = \frac{1}{g^2 q}$ and c is the normalization constant for this pdf.

Actual experience in developing an EKF for angle-only Reentry Vehicle (RV) tracking via jammed (range-denied) radar using triangulation (RVTRIANG), as modified from an earlier EKF for tracking RV's via unjammed radar, convinced me that such goals are best carried out in specific well thought out stages. Examples are, first, for a constant gravity, then for inverse-squared gravity. First, for a non-rotating earth, then for a rotating earth. More detail on this aspect is provided in [20, footnotes 5 and 8]. A representative plan for EKF development that I adhered to for this endeavor is depicted in Table 3.

Table 3. Stages and confirmation tests for phased EKF development.

STAGE	SOFTWARE UTILIZED	COVARIANCE ANALYSIS	FILTER PORTION	LINEARIZATION	RV TRAJECTORY GENERATOR	PURPOSE
1	RVTRIANG	Covariance only.	None	About true trajectory.	Straight line.	Establish benchmark for later comparisons.
2	Modified RVTRIANG (copy 1)	Introduce EKF Covariance mechanization (except linearized about true states).	Introduce EKF Filter portion (except linearized about true states). Print/plot states.	Same as above (so not a true EKF yet).	Same as above. Need to print/plot states.	See if covariances are similar to above case. See if filter estimates follow true trajectory (maybe with lag).
3	Modified RVTRIANG (copy 2)	Introduce linearization about estimates (so now a true EKF).	Same as above.	Introduce linearization about estimates (so now a true EKF).	Same as above.	See how closely EKF estimates follow true trajectory.
4	Modified RVTRIANG (copy 3)	Same as above.	Same as above.	Same as above.	Introduce nonlinear equations for conic RV trajectory.	See how closely EKF estimates follow true trajectory.
5	Modified RVTRIANG (copy 4)	Same as above.	Same as above.	Introduce relinearization.	Same as above.	See improvement in how closely EKF estimates follow true trajectory.
6	Modify EXEC to conform to goal of 3 main modules.	Same as above.	Same as above.	Same as above.	Keep separate.	See that answers and EKF outputs are the same as above case.

§3 Appendix: A closed-form analytic solution – useful for testing Kalman filter covariance calculation

The steady-state covariance solution before and after a measurement update (for periodic measurement usage) corresponds to solving the following two familiar KF mechanization equations for $\tilde{P}$ and $\hat{P}$, respectively, being:

$$\tilde{P} - BQB^{\top} = \Phi(I - GC)\tilde{P}\Phi^{\top},$$
$$\hat{P} = (I - GC)\tilde{P}.$$

While it would be desirable to just pluck the steady-state solution from [26], it turns out to *not* be quite that simple and easy. As laid out in [26] after laboring through a lot of algebra and parameter scaling, we have to solve a biquartic equation [26, Eq.(A9)] as an intermediate calculation. In order to make this challenge somewhat easier, instead of first specifying σ_a and σ_x in equations (24) and (25) beforehand, the new contribution provided here is to use a trick of convenience by finding the value that makes the following biquartic easy to solve for S:

$$S^4 - 6S^3 + 10S^2 - 6(1 + 2r^2)S + (1 + 3r^2) = 0\,, \tag{26}$$

where we recall that while quadratic equations are easy to solve, general cubics and quartics/biquartics are extremely challenging and messy in general. The trick is to force a convenient answer, as say,

$$S = 6 \tag{27}$$

to be a solution of equation (26) [26, Eq.(A9)] by choosing the value of r (appearing in equation (26)) for convenience. This proper value of r can be selected by first performing the following division exercise:

$$\begin{array}{l|l|l|l|l}
 & S^3 & +10S & +(54-12r^2) & \text{remainder}: \quad +325-69r^2 \\
\hline
S-6 & S^4 & -6S^3 & +10S^2 & -6(1+2r^2)S \quad +(1+3r^2) \\
 & S^4 & -6S^3 & & \\
\hline
 & & 10S^2 & -6(1+2r^2)S & \\
 & & 10S^2 & -60S & \\
\hline
 & & & (54-12r^2)S & +(1+3r^2) \\
 & & & (54-12r^2)S & -6(54-12r^2) \\
\hline
 & & & & 1+3r^2+324-72r^2
\end{array} \tag{28}$$

So $S = 6$ is a root of equation (26) if the remainder in the above is zero as

$$325 - 69r^2 = 0 \tag{29}$$

or

$$r^2 = \frac{325}{69} \Rightarrow r = \sqrt{\frac{325}{69}} = 2.17028\,. \tag{30}$$

Here, we remark that arbitrary solutions of equation (26) can't be forced (as in seeking to make $S = 2$ be a solution) because the remainder term will correspond to an "imaginary" value for r, which needs to be a real variable to be viable in this application.

Now from the equation following equation (15) in [26], we have that

$$r \triangleq \frac{12\sigma_x}{\sigma_a T^3}\,. \tag{31}$$

From equation (30) above,

$$2.17028 = r = \frac{12\sigma_x}{\sigma_a T^3}\,, \tag{32}$$

we can now take

$$T = 2 \tag{33}$$

and

$$\sigma_x = 4\,, \tag{34}$$

so that rearranging equation (32) with these two assignments of equations (33) and (34) yields

$$\sigma_a = \frac{12(4)}{2.17028(8)} = \frac{6}{2.17028} = 2.764620 \tag{35}$$

or, referring back to equations (24) and (25), yields the following two specifications

$$Q_1 = \sigma_a^2 = 7.643125 \tag{36}$$

and

$$R_1 = \sigma_x^2 = (4)^2 = 16 \tag{37}$$

that are necessary to be pinned-down for a well-posed KF. According to [26, prior to Fig.1], the dimensionless quantity r defined in Eq. 31 can be interpreted as a type of noise-to-signal ratio.

Now, following equation (17) of [26],

$$S_1 \triangleq \sqrt{S^2 + r^2} = \sqrt{36 + \frac{325}{69}} = 6.38045, \tag{38}$$

$$S_2 \triangleq \sqrt{4S - 1} = \sqrt{24 - 1} = \sqrt{23} = 4.7958, \tag{39}$$

$$A \triangleq \sqrt{3} = 1.73205, \tag{40}$$

$$\begin{aligned} D \triangleq & r\sqrt{\frac{1}{2} + r^2\left(\frac{145}{16} + r^2\left(\frac{97}{2} + 81r^2\right)\right)} \\ = & 2.17028\sqrt{\frac{1}{2} + \frac{325}{69}\left(\frac{145}{16} + \frac{325}{69}\left(\frac{97}{2} + 81\frac{325}{69}\right)\right)} \\ = & 2.17028\sqrt{\frac{1}{2} + 9582.91} \\ = & 2.17028(97.8949) = 212.46, \end{aligned} \tag{41}$$

$$\begin{aligned} C \triangleq & r^2\left(\frac{17}{4} - 9r^2\right) + \frac{1}{27} = \frac{325}{69}\left(\frac{17}{4} - 9\frac{325}{69}\right) + \frac{1}{27} \\ = & -179.651 + \frac{1}{27} = -179.614, \end{aligned} \tag{42}$$

$$U \triangleq (D - C)^{\frac{1}{3}} = (212.46 + 179.614)^{1/3} = 7.3190, \tag{43}$$

$$V \triangleq (D + C)^{1/3} = (212.46 - 179.614)^{1/3} = 3.2025, \tag{44}$$

$$\begin{aligned} Z \triangleq & U - V + \frac{5}{3} = 7.3190 - 3.2025 + \frac{5}{3} \\ = & 4.1165 + 1.66666 = 5.78317, \end{aligned} \tag{45}$$

$$\begin{aligned} b \triangleq & Z - \sqrt{Z^2 - 3r^2 - 1} \\ = & 5.8 - \sqrt{(5.8)^2 - 3\frac{325}{69} - 1} = 5.8 - \sqrt{46.73} = 5.8 - 6.8360 \\ = & -1.0361, \end{aligned} \tag{46}$$

$$a \triangleq 3 + \sqrt{3Z - 1} = 3 + \sqrt{3(5.8) - 1} = 3 + 4.04969 = 7.04969. \tag{47}$$

From equation (18) of [26],

$$\hat{Y}_{13} = \frac{(S_1 - S)}{r} = \frac{6.38045 - 6}{\sqrt{\frac{325}{69}}} = \frac{0.38045}{2.17028} = 0.17529, \tag{48}$$

$$\hat{Y}_{11} = \frac{2S\hat{Y}_{13}}{r} = \frac{12(0.17529)}{r} = 0.968096, \tag{49}$$

$$\hat{Y}_{12} = \frac{S_2\hat{Y}_{13}}{A} = \frac{4.7958(0.17529)}{\sqrt{3}} = 0.485352, \tag{50}$$

$$\begin{aligned} \hat{Y}_{22} = & \frac{S_2(3 + 2S - AS_2)}{2A} - S_1 \\ = & \frac{4.7958[3 + 12 - \sqrt{3}(4.7958)]}{2\sqrt{3}} - 6.38045 \\ = & 9.266584 - 6.38045 = 2.8861, \end{aligned} \tag{51}$$

$$\hat{Y}_{33} = \frac{(S_2 - A)}{2A} = \frac{4.7958 - \sqrt{3}}{2\sqrt{3}} = 0.88443, \tag{52}$$

$$\hat{Y}_{23} = \hat{Y}_{33}^2 = 0.7822216\,. \tag{53}$$

We remark that the expression used here in equation (53) is my correction, which has been gracefully acknowledged as being correct by Ramachandra in personal correspondence. Similarly from [26, Eq.(17)], we have that

$$\tilde{Y}_{13} = \frac{(S_1 + S)}{r} = \frac{6.38045 + 6}{\sqrt{\frac{325}{69}}} = 5.7045, \tag{54}$$

$$\tilde{Y}_{11} = \frac{2S\tilde{Y}_{13}}{r} = 31.541577, \tag{55}$$

$$\tilde{Y}_{12} = \frac{S_2\tilde{Y}_{13}}{A} = \frac{4.7958(5.7045)}{\sqrt{3}} = 15.7950, \tag{56}$$

$$\tilde{Y}_{22} = \frac{S_2(3 + 2S + AS_2)}{2A} - S_1$$

$$=\frac{4.7958[3+12+\sqrt{3}(4.7958)]}{2\sqrt{3}}-6.38045$$
$$=25.885822, \tag{57}$$
$$\tilde{Y}_{33}=\frac{(S_2+A)}{2A}=\frac{4.7958+\sqrt{3}}{2\sqrt{3}}=1.884428, \tag{58}$$
$$\tilde{Y}_{23}=\tilde{Y}_{33}^2=3.55107. \tag{59}$$

We can now specify the steady-state KF covariances $\tilde{P}$ and $\hat{P}$, respectively, since we have already evaluated all the $\tilde{Y}_{ij}$'s and $\hat{Y}_{ij}$'s that are necessary intermediate calculations. Using the scalings in [26, Eq.(16)] to unravel the $\tilde{P}$'s from the $\tilde{Y}$'s ([26, Eqs.(48-53)]) yields

$$\tilde{P}_{11}=\sigma_x^2\tilde{Y}_{11}=16(31.541577)=504.6652, \tag{60}$$
$$\tilde{P}_{12}=\frac{1}{2}\sigma_x\sigma_a T^2\tilde{Y}_{12}=\frac{16(2.764620)}{2}(15.79500)=349.33738, \tag{61}$$
$$\tilde{P}_{13}=\sigma_x\sigma_a T\tilde{Y}_{13}=4(2.764620)2(5.704522)=126.1666, \tag{62}$$
$$\tilde{P}_{22}=\frac{\sigma_a^2T^4}{12}\tilde{Y}_{22}=\frac{(2.764620)^2 16}{12}(25.885822)=263.79805, \tag{63}$$
$$\tilde{P}_{23}=\frac{\sigma_a^2T^3}{2}\tilde{Y}_{23}=\frac{(2.764620)^2 8}{2}(3.551069)=108.5650, \tag{64}$$
$$\tilde{P}_{33}=\sigma_a^2T^2\tilde{Y}_{33}=(2.764620)^2 4(1.884428)=57.6117. \tag{65}$$

Thus, the 3×3 steady-state covariance prior to one of the periodic measurement updates is

$$\tilde{P}_1=\begin{bmatrix}504.6652 & 349.3374 & 126.1666\\ \cdot & 263.79805 & 108.5650\\ \cdot & \cdot & 57.6117\end{bmatrix}, \tag{66}$$

which is a symmetric partition of the symmetric 6×6 matrix

$$\tilde{P}_2=\begin{bmatrix}\tilde{P}_1 & \vdots & 0\\ \cdots & \cdots & \cdots\\ 0 & \vdots & \tilde{P}_1\end{bmatrix} \tag{67}$$

to be used as an explicit check on the output generated by the software implementation under test.

Using the scalings in [26, Eq.(16)] which can also be used in an identical fashion as just demonstrated above to also unravel the $\hat{P}$'s from the $\hat{Y}$'s ([26, Eqs.(42-47)]) to yield

$$\hat{P}_{11}=\sigma_x^2\hat{Y}_{11}=16(0.969269)=15.5083, \tag{68}$$

$$\hat{P}_{12}=\frac{1}{2}\sigma_x\sigma_a T^2\hat{Y}_{12}=8(2.764620)(0.485352)=10.7345, \tag{69}$$
$$\hat{P}_{13}=\sigma_x\sigma_a T\hat{Y}_{13}=8(2.764620)(0.1753)=3.8771, \tag{70}$$
$$\hat{P}_{22}=\frac{\sigma_a^2T^4}{12}\hat{Y}_{22}=\frac{(2.764620)^2 16}{12}(2.8861)=29.4118, \tag{71}$$
$$\hat{P}_{23}=\frac{\sigma_a^2T^3}{2}\hat{Y}_{23}=(2.764620)^2 4(0.782216)=23.9143, \tag{72}$$
$$\hat{P}_{33}=\sigma_a^2T^2\hat{Y}_{33}=(2.764620)^2 4(0.88443)=27.0392. \tag{73}$$

Thus, the 3×3 steady-state covariance immediately after one of the periodic measurement updates is

$$\hat{P}_1=\begin{bmatrix}15.5083 & 10.7345 & 3.8771\\ \cdot & 29.4118 & 23.9143\\ \cdot & \cdot & 27.0392\end{bmatrix}, \tag{74}$$

which is a symmetric partition of the symmetric 6×6 matrix

$$\hat{P}_2=\begin{bmatrix}\hat{P}_1 & \vdots & 0\\ \cdots & \cdots & \cdots\\ 0 & \vdots & \hat{P}_1\end{bmatrix} \tag{75}$$

to be used as an explicit test case check on the software implementation. The obvious sanity checks between $\tilde{P}_1$ and $\hat{P}_1$ are satisfied since the latter covariance representing the uncertainty in the three states of position, velocity, and acceleration after a measurement update are in fact all smaller, as expected. The three cross-check equations following equation (19) of [26] for the main diagonal entries of both $\tilde{P}_1$ and $\hat{P}_1$ are satisfied as

$$(\tilde{P}_{22}-\hat{P}_{22})=\sigma_a^2T^4\frac{(4S-1)}{12},$$

or

$$263.798-\hat{P}_{22}=(2.764620)^2\frac{16(24-1)}{12}=234.389, \tag{76}$$

or

$$\hat{P}_{22}=29.408871,$$

which agrees with the result of equation (71) so $\tilde{P}_{22}$ and $\hat{P}_{22}$ are consistent with this cross-check;

$$(\tilde{P}_{33}-\hat{P}_{33})=\sigma_a^2T^2,$$

or

$$57.6117-\hat{P}_{33}=(2.764620)^2 4=30.5724, \tag{77}$$

or

$$\hat{P}_{33} = 27.0392,$$

which agrees with the result of equation (73) so $\tilde{P}_{33}$ and $\hat{P}_{33}$ are consistent with this cross-check;

$$(\tilde{P}_{11} - \hat{P}_{11}) = \sigma_x^2 \frac{4S^2}{r^2},$$

or

$$504.6652 - \hat{P}_{11} = 16\frac{4(36)}{325/69} = 489.1569, \tag{78}$$

or

$$\hat{P}_{11} = 15.5083,$$

which agrees with the result of equation (68) so $\tilde{P}_{11}$ and $\hat{P}_{11}$ are consistent with this cross-check. Now this closed-form example test case can be used to check-out and confirm output from any software implementation of a discrete-time KF covariance calculation for the parameters of Section 2.2.5 or augmentation of Section 2.2.3 to any state size that is an integer multiple of the original 3 states.

References

1. Bar-Shalom, Y. and T. E. Fortmann, *Tracking and Data Association*, Academic Press, New York, 1988.
2. Benes, V. E., Exact finite-dimensional filters for certain diffusions with nonlinear drift, *Stochastics* **5** (1981), 65-92.
3. Bierman, G. J., Numerical experience with modern estimation algorithms, *Proc. of IEEE Conf. on Decis. Contr.*, Ft. Lauderdale, FL, Dec., 1985, 1896-1901.
4. Boozer, D. D. and W. L. McDaniel, On innovation sequence testing of the Kalman filter, *IEEE Trans. on Auto. Contr.* **17** (1972), 158-160.
5. Chen, G., Convergence analysis for inexact mechanization of Kalman filters, *IEEE Trans. on Aero. Electr. Sys.* **28** (1992), 612-621.
6. Daum, F. E., Exact finite-dimensional nonlinear filters, *IEEE Trans. on Auto. Contr.* **31** (1986), 616-622.
7. Daum, F. E., Solution of the Zakai equation by separation of variables, *IEEE Trans. on Auto. Contr.* **32** (1987), 941-943.
8. Gelb, A. (ed.), *Applied Optimal Estimation*, M.I.T. Press, Cambridge, MA, 1974.
9. Hall, E. B. and G. L. Wise, An analysis of convergence problems in Kalman filtering which arise from numerical effects, *Proc. of the 33rd Midwest Symp. on Circ. Sys.*, Calgary, Alberta, Canada, Aug., 1990.
10. Jazwinski, A. H., *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.

11. Kerr, T. H., An invalid norm appearing in control and estimation, *IEEE Trans. on Auto. Contr.* **23** (1978), 73-74 (Correction on pp. 1117-1118, Dec. 1978).
12. Kerr, T. H., Decentralized filtering and redundancy management for multisensor navigation, *IEEE Trans. on Aero. Electr. Sys.* **23** (1987), 83-119 (minor corrections appear on p. 412 of May and p. 599 of July 1987 issues of the same journal).
13. Kerr, T. H., Testing matrices for definiteness and application examples that spawn the need, *AIAA J. of Guid. Contr. Dynam.* **10** (1987), 503-506 (reply to and rebuttal by author in **12** (1989), 767).
14. Kerr, T. H., Computational techniques for the matrix pseudoinverse in minimum variance reduced-order filtering and control, in *Control and Dynamic Systems-Advances in Theory and Applications*, Vol. 28: *Advances in Algorithms and Computational Techniques for Dynamic Control Systems*, Part 1 of 3, C. T. Leondes (ed.), Academic Press, New York, 1988, 57-107.
15. Kerr, T. H., Rationale for Monte-Carlo simulator design to support multichannel spectral estimation and/or Kalman filter performance testing and software validation/verification using closed-form test cases, M.I.T Lincoln Laboratory Report No. PA-512, Lexington, MA, 22 Dec. 1989.
16. Kerr, T. H., An analytic example of a Schweppe likelihood ratio detector, *IEEE Trans. on Aero. Electr. Sys.* **25** (1989), 545-558.
17. Kerr, T. H., A constructive use of idempotent matrices to validate linear systems analysis software, *IEEE Trans. on Aero. Electr. Sys.* **26** (1990), 935-952 (minor corrections in the same journal **27** (1991), 951-952.
18. Kerr, T. H., On misstatements of the test for positive semidefinite matrices, *AIAA J. of Guid. Contr. Dynam.* **13** (1990), 571-572.
19. Kerr, T. H., Fallacies in computational testing of matrix positive definiteness/semidefiniteness, *IEEE Trans. on Aero. Electr. Sys.* **26** (1990), 415-421.
20. Kerr, T. H., Streamlining measurement iteration for EKF target tracking, *IEEE Trans. on Aero. Electr. Sys.* **27** (1991), 408-420 (minor correction appears in Nov. 1991 issue).
21. Kwakernaak, H. and R. Sivan, *Linear Optimal Control Systems*, Wiley-Inter-Science, New York, 1972.
22. Lamperti, J., *Probability: A Survey of the Mathematical Theory*, Benjamin, Inc., New York, 1966.
23. Martin, W. C. and A. R. Stubberud, An additional requirement for innovations testing in system identification, *IEEE Trans. on Auto. Contr.* **19** (1974), 583-585.
24. Maybeck, P. S., *Stochastic Models, Estimation, and Control*, Vol.1, Academic Press, New York, 1979.
25. Patel, J. K., C. H. Kapadia, and D. B. Owens, *Handbook of Statistical Distributions*, Marcel Dekker, New York, 1976.
26. Ramachandra, K. V., Optimum steady state position, velocity, and acceleration

estimation using noisy sampled position data, *IEEE Trans. on Aero. Electr. Sys.* **23** (1987), 705-708 (a correction appears in the May issue of 1988).
27. Sanghai-IAM, S. and T. E. Bullock, Analysis of discrete-time Kalman filtering under incorrect noise covariances, *IEEE Trans. on Auto. Contr.* **35** (1990), 1304-1309.
28. Tam, L.-F., W.-S. Wong, and S. S.-T. Yau, On a necessary and sufficient condition for finite dimensionality of estimation algebras, *SIAM J. on Contr. Optim.* **28** (1990), 173-185.
29. *Kalman Filtering Software: User's Guide*, Optimization Software, Inc., New York, 1984.
30. Special issue of *IEEE Trans. on Auto. Contr.* **28** (3) (1983), on *Applications of Kalman Filters.*

Thomas H. Kerr
Lincoln Laboratory of M.I.T.
and TeK Associates
11 Paul Revere Road
Lexington, MA 02173-6632

Further Reading

1. Anderson, B. D. O. and J. B. Moore, *Optimal Filtering*, Prentice-Hall, Englewood Cliffs, N. J., 1979.
2. Aoki, M., *Optimization of Stochastic Systems: Topics in Discrete-Time Dynamics*, Academic Press, New York, 1989.
3. Balakrishnan, A. V., *Kalman Filtering Theory*, Optimization Software, Inc., New York, 1984 (1st ed.) and 1987 (2nd ed.).
4. Bozic, S. M., *Digital and Kalman Filtering*, John Wiley & Sons, New York, 1979.
5. Brammer, K. and G. Sifflin, *Kalman-Bucy Filters*, Artech House Inc., Boston, 1989.
6. Brown R. G. and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley & Sons, New York, 1992.
7. Bucy, R. S. and P. D. Joseph, *Filtering for Stochastic Processes with Applications to Guidance*, John Wiley & Sons, New York, 1968.
8. Caines, P. E., *Linear Stochastic Systems*, John Wiley & Sons, New York, 1988.
9. Catlin, D. E., *Estimation, Control, and Discrete Kalman Filters*, Springer-Verlag, New York, 1989.
10. Chen, H. F., *Recursive Estimation and Control for Stochastic Systems*, John Wiley & Sons, New York, 1985.
11. Chui, C. K. and G. Chen, *Kalman Filtering with Real-Time Applications*, Springer-Verlag, New York, 1987 (1st ed.) and 1991 (2nd ed.)
12. Davis, M. H. A., *Linear Estimation and Stochastic Control*, John Wiley & Sons, New York, 1977.
13. Gelb, A., *Applied Estimation Theory*, M.I.T. Press, Cambridge, MA, 1974.
14. Goodwin, G. C. and K. S. Sin, *Adaptive Filtering Prediction and Control*, Prentice-Hall, Englewood Cliffs, N. J., 1984.
15. Haykin, S., *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs, N. J., 1986.
16. Jazwinski, A. H., *Stochastic Processes and Filtering Theory*, Academic Press, New York, 1970.
17. Kailath, T., *Lectures on Wiener and Kalman Filtering*, CISM Courses and Lectures, No.40, Springer-Verlag, New York, 1981.
18. Kailath, T., *Course Notes on Linear Estimation*, Stanford University, Stanford, CA, 1982.
19. Kallianpur, G., *Stochastic Filtering Theory*, Springer-Verlag, New York, 1980.

20. Krishnan, V., *Nonlinear Filtering and Smoothing: An Introduction to Martingales, Stochastic Integrals, and Estimation*, Jonh Wiley & Sons, New York, 1984.
21. Kumar, P. R. and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*, Prentice-Hall, Englewood Cliffs, N. J., 1986.
22. Leondes, C. T. (ed.), *Theory and Applications of Kalman Filtering*, AGARDograph No.139, Technical Editing and Reproduction Ltd., London, 1970.
23. Lewis, F. L., *Optimal Estimation*, John Wiley & Sons, New York, 1986
24. Maybeck, P. S., *Stochastic Models, Estimation, and Control*, Vols. 1-3, Academic Press, New York, 1979.
25. Mendel, J. M., *Lessons in Digital Estimation Theory*, Prentice-Hall, Englewood Cliffs, N. J., 1987.
26. Otter, P. W., *Dynamic Feature Space Modeling, Filtering and Self-Tuning Control of Stochastic Systems*, Springer-Verlag, New York, 1985.
27. Ruymgaart, P. A. and T. T. Soong, *Mathematics of Kalman-Bucy Filtering*, Springer-Verlag, New York, 1985 (1st ed.) and 1988 (2nd ed.).
28. Sorenson, H. W. (ed.), *Kalman Filtering: Theory and Application*, IEEE Press, New York, 1985.
29. Strobach, P., *Linear Prediction Theory: A Mathematical Basis for Adaptive Systems*, Springer-Verlag, New York, 1990.
30. Young, P., *Recursive Estimation and Time-Series Analysis*, Springer-Verlag, New York, 1984.

Notation

A^+	pseudo-inverse of matrix A
A, $A(t)$, A_k	$n \times n$ system matrices
$A^{1/2}$	"square-root" of A
B, $B(t)$, B_k	$p \times n$ control input matrices
C, $C(t)$, C_k	$q \times n$ measurement matrices
$Cov\,(X, Y)$	covariance of random variables X and Y
$E\{X\}$	expectation of random variable X
$E\{X\|Y\}$	conditional expectation of X given Y
G	steady-state Kalman gain matrix
G_k	Kalman gain matrix
I_n	$n \times n$ identity matrix
$N(m, \sigma^2)$	Gaussian distribution with mean m and variance σ^2
$P_{k,k}$	error covariance matrix: $Cov(\hat{\mathbf{x}}_{k,k} - \mathbf{x}_{k,k})$
$P_{k,k-1}$	error covariance matrix: $Cov(\hat{\mathbf{x}}_{k,k-1} - \mathbf{x}_{k,k-1})$
$p(x)$	probability density function
$p(x_1, x_2)$	joint probability density function
$p(x_1\|x_2)$	conditional probability density function
Q_k	covariance matrix of system random vector $\underline{\xi}_k$
R_k	covariance matrix of measurement random vector $\underline{\eta}_k$
S_k	covariance matrix of system and measurement vectors
tr	trace of a matrix
$\mathbf{u}_k$	deterministic control input (at the kth time instant)
$Var\,(X)$	variance of random variable X
$Var\,(X\|Y)$	conditional variance of X given Y
$\mathbf{v}_k$	observation (or measurement) data (at the kth time instant)
W_k	weight (matrix) (at the kth time instant)
w_k	weight (scalar) (at the kth time instant)
$\mathbf{x}_k$	state vector (at the kth time instant)
$\hat{\mathbf{x}}_k$, $\hat{\mathbf{x}}_{k,k}$	optimal filtering estimate of $\mathbf{x}_k$
$\hat{\mathbf{x}}_{k,k-1}$	one-step ahead optimal prediction of $\mathbf{x}_k$
$\langle \mathbf{x}, \mathbf{y} \rangle$	inner product of $\mathbf{x}$ and $\mathbf{y}$
$\mathbf{x}^\top$, $\Gamma^\top$	transposes of vector $\mathbf{x}$ and matrix Γ
δ_{ij}	Kronecker delta

$\underline{\eta}_k$	measurement noise (at the kth time instant)
$\underline{\xi}_k$	system noise (at the kth time instant)
$\Phi_{i,j}$	transition matrix (from the jth to the ith state)
0	zero (number, vector, matrix, function)

Subject Index

Adaptive filter 71,87
  adaptive Kalman filter 65,98,100
Bayesian estimate 23,66
  Bayesian inference scheme 23
Bellman-Gronwall lemma 179,184
Bias 15
Bounding ellipsoidal set 163,171
Central limit theorem 195
Cholesky decomposition 43
Coloured noise 96,105
Confidence value 161
Convex propagation theorem 144
Convex representation theorem 145
Cramer-Rao lower bound 10
Credal convexity 140
  Credal probability 142
  Credal state 142
Degree of boldness 151
Distributed filtering 161
Epistemic utility 140
  maximum expected 150,151
Estimate 24,26,28,42,67,68,70,140
  a priori 7,9,15,66,87,157
  a posteriori 7,9,16,66,142
  conditional mode 67
  correlation method 68
  covariance-matching technique 70
  Fisher 24
    classical 26
    generalized 28
  maximum likelihood (ML) 42,67
    joint 67
    marginal 67
  set-valued 140
Least squares 43,44
Extended Kalman filter (EKF) 3,15,211
  ideal (IEKF) 9,10
  iterated 13
  modified (MEKF,MGEKF) 9,11,39
  standard (EKF,SEKF) 4
Fisher initialization 23
Fixed point smoother (FPS) 56
Fokker-Planck equation 7
Gaussian noise 65,89
  Gaussian analogue 120
  Gaussian sum 113
  non-Gaussian noise 113,161
Generalized Fisher error covariance 28
Hassian (matrix) 16
Kalman extrapolation equation 25
Kalman filter 3,6,15,23,25,39,65,87,113,
    114,139,161,179,193
  adaptive 65,98,100
  class B JTIDS filter 194
  diffuse (DKF) 39,45
  extended (EKF) 3,15
  GPF filter 194
  IV&V Kalman filter code 193
  MFBARS filter 194
  on-line Kalman filtering 90
  PTA filter 194
  PINS filter 194
  robust adaptive Kalman filtering 65
  robust 181
  RV tracking filter 194
  set-valued Kalman filtering 139,146
  SINS correction filter 194
  STAR filter 194
  suboptimal Kalman filtering 113,114
Numerical approximation 193