Tracking and Kalman Filtering Made Easy. Eli Brookner

ISBNs: 0-471-18407-1 (Hardback); 0-471-22419-7 (Electronic)

2

KALMAN FILTER

2.1 TWO-STATE KALMAN FILTER

Up to now we have used a *deterministic* description for the target motion. Specifically, we have assumed a target having a constant-velocity motion as given by

$$x_{n+1} = x_n + T\dot{x}_n \tag{1.1-1a}$$

$$\dot{x}_{n+1} = \dot{x}_n \tag{1.1-1b}$$

In the real world the target *will not have a constant velocity for all time*. There is actually uncertainty in the target trajectory, the target accelerating or turning at any given time. Kalman allowed for this uncertainty in the target motion by adding a random component to the target dynamics [19, 20]. For example, a random component u_n could be added to the target velocity as indicated by the following equations for the target dynamics:

$$x_{n+1} = x_n + T\dot{x}_n \tag{2.1-1a}$$

$$\dot{x}_{n+1} = \dot{x}_n + u_n \tag{2.1-1b}$$

where u_n is a random change in velocity from time n to time $n+1$. We assume u_n is independent from n to $n+1$ for all n and that it has a variance σ_u^2. Physically u_n represents a random-velocity jump occurring just prior to the $n+1$ observation.

We now have a system dynamics model with some randomness. This model is called the constant-velocity trajectory model with a random-walk velocity.

The random-velocity component u_n is sized to account for a possible target acceleration or unexpected target turn. The random dynamics model component u_n in the literature goes by the names process noise [6, 30], plant noise [8, 29, 30], driving noise [5, 8], dynamics noise [119], model noise, and system noise (see Appendix).

Let x_{n+1} represent the true location of the target at time $n+1$. Let $x^*_{n+1,n}$ represent an estimated predicted position of the target at time $n+1$ based on the measurements made up to and including time n. Kalman addressed the question of finding the optimum estimate among the class of all linear and nonlinear estimates that minimizes the mean square error

$$\left(x^*_{n+1,n} - x_{n+1}\right)^2 \tag{2.1-2}$$

After much effort Kalman found that the optimum filter is given by the equations

$$\dot{x}^*_{n+1,n} = \dot{x}^*_{n,n-1} + \frac{h_n}{T}\left(y_n - x^*_{n,n-1}\right) \tag{2.1-3a}$$

$$x^*_{n+1,n} = x^*_{n,n-1} + T\dot{x}^*_{n+1,n} + g_n\left(y_n - x^*_{n,n-1}\right) \tag{2.1-3b}$$

But these are identical to the g–h filter given previously, specifically (1.2-11). For the Kalman filter the weights g_n and h_n depend on n. Furthermore, as shall be seen later, g_n and h_n are functions of the variance of the radar position measurement, that is, the variance of ν_n; see (1.2-17). These filter constants are also a function of the accuracy to which we know the position and velocity before any measurements are made, that is, of our prior knowledge of the target trajectory, as given by the a priori variance of the target position and velocity. (Such information might be available when the track is being started after handoff from another sensor.) In the steady state the filter constants g_n and h_n are given by [12]

$$h = \frac{g^2}{2-g} \tag{2.1-4}$$

This equation is identical to that for the Benedict–Bordner filter given by Eq. (1.2-27). Thus the steady-state Kalman filter is identical to the Benedict–Bordner g–h filter. The more general Kalman filter was developed before the Benedict–Bordner filter. The Kalman filter was first published in 1960 [19] while the Benedict–Bordner filter was published in 1962 [10]. In the literature, when the g–h Benedict–Bordner filter is derived as a steady-state g–h Kalman filter as described above, it is usually referred to as the optimum g–h filter [14, 15].

The Kalman filter has the advantage that it does not require the use of the ad hoc equation relating the rms target prediction error to the g–h bias error

as given by (1.2-22) and the bias equation as given by (1.2-15) to determine the tracking-filter update period T as done in Example 1 Section 1.2.7. Instead for the Kalman filter the update period is obtained using the equation [12]

$$T^2 \frac{\sigma_u^2}{\sigma_x^2} = \frac{g^4}{(2-g)^2(1-g)} = \frac{h^2}{1-g} \tag{2.1-5}$$

which relates the target update period to the variance of the target dynamics σ_u^2 as well as to the noise measurement error and the filter parameter g. [See problem 2.4-1 for derivation of (2.1-4) and (2.1-5).] Figure 1.2-7 to 1.2-9 were actually developed for optimum g–h filter in Reference 11. However, in developing these figures, (2.1-5) is not used, only (2.1-4).

It remains to determine how to specify the variance of the target dynamics σ_u^2. Let the maximum expected target acceleration be $\ddot{x}_{\max}$. Then the greatest change in velocity in one sample period T is $T\ddot{x}_{\max}$ and σ_u is chosen to be given by [12]

$$\sigma_u = \frac{T\ddot{x}_{\max}}{B} \tag{2.1-6}$$

where B is a constant. A good value for B is 1 when tracking ballistic targets [12]. With this choice of B the errors $3\sigma_{n+1,n}$ and b^* will be about equal. For maneuvering targets a larger B may be better [12]. If the maneuver were independent from sample to sample, then $B = 3$ would be suitable [12].

Using (1.2-15), (2.1-6), and (2.1-5), the following expression for the normalized bias error b^* is obtained for the g–h Kalman filter in steady state [12]:

$$\frac{b^*}{\sigma_x} = \frac{B}{\sqrt{1-g}} \tag{2.1-7}$$

2.2 REASONS FOR USING THE KALMAN FILTER

Since the steady-state Kalman filter is identical to the Benedict–Bordner filter, the question arises as to why we should use the Kalman filter. The benefits accrued by using the Kalman filter are summarized in Table 2.2-1. First, while in the process of computing the filter weights g_n and h_n for the Kalman filter, calculations of the accuracy of the Kalman filter predictions are made. This prediction information is needed for a weapon delivery system to determine if the predicted position of the target is known accurately enough for a target kill. It is also needed to accurately predict where a detected SCUD, intermediate-

TABLE 2.2-1. Benefits of Kalman Filter

Provides running measure of accuracy of predicted position needed for weapon kill probability calculations; impact point prediction calculation
Permits optimum handling of measurements of accuracy that varies with n; missed measurements; nonequal times between measurements
Allows optimum use of a priori information if available
Permits target dynamics to be used directly to optimize filter parameters
Addition of random-velocity variable, which forces Kalman filter to be always stable

range ballistic missile (IRBM), or intercontinental ballistic missile (ICBM) would land. It makes a difference whether the SCUD, IRBM, or ICBM is landing in neutral territory such as the ocean or in a major city. It is also needed for determining where an artillery shell, mortar shell, SCUD, IRBM, or ICBM was launched. In the cases of the mortar shell, artillery shell, and SCUD this information is needed in order to destroy the launcher or canon. In the case of the IRBM and ICBM this information is needed to determine who is firing so that the appropriate action can be taken. One does not want to take action against country A when it is country B that is firing. The Kalman filter allows one to make estimates of the launcher location and estimate the accuracy of these estimates.

The Kalman filter makes optimal use of the target measurments by adjusting the filter weights g_n and h_n to take into account the accuracy of the nth measurement. For example, if on the nth measurement the signal-to-noise ratio (SNR) is very good so that a very accurate target position measurement is obtained, then g_n and h_n are automatically adjusted to take this into account. Specifically, they are made larger so as to give more weight to this more accurate measurement. If the target had been missed on the $(n-1)$st look, then g_n and h_n are optimally adjusted to account for this. The filter parameters g_n and h_n are also adjusted to allow for nonequal times between measurements.

The Kalman filter optimally makes use of a priori information. Such a priori information could come from another radar that had been previously tracking the target and from which the target is being handed over—such as handover from a search to tracking radar or from one air route surveillance radar (ARSR) to another. The data from the other radar can be used to optimally set up the Kalman g–h filter for the new radar. The Kalman filter automatically chooses weights that start with large g and h values as needed for optimum track initiation. The weights slowly transition to a set of small constant g's and h's after track initiation. The target dynamics model incorporated by the Kalman filter allows direct determination of the filter update rate by the use of (2.1-5). Finally the addition of the random-velocity variable u_n forces the Kalman filter to always be stable.

2.3 PROPERTIES OF KALMAN FILTER

We will now give some physical feel for why the Kalman filter is optimum. Let us go back to our discussion in Section 1.2. Recall that for our two-state *g–h* tracking we have at time n two estimates of the target position. The first is y_n, based on the measurement made at time n (see Figure 1.2-3). The second is the prediction $x^*_{n,n-1}$, based on past measurements. The Kalman filter combines these two estimates to provide a filtered estimate $x^*_{n,n}$ for the position of the target at time n. The Kalman filter combines these two estimates so as to obtain an estimate that has a minimum variance, that is, the best accuracy. The estimate $x^*_{n,n}$ will have a minimum variance if it is given by [5–7]

$$x^*_{n,n} = \left[\frac{x^*_{n,n-1}}{\mathrm{VAR}(x^*_{n,n-1})} + \frac{y_n}{\mathrm{VAR}(Y_n)}\right] \frac{1}{1/\mathrm{VAR}(x^*_{n,n-1}) + 1/\mathrm{VAR}(y_n)} \tag{2.3-1}$$

That (2.3-1) provides a good combined estimate can be seen by examining some special cases. First consider the case where y_n and $x^*_{n,n-1}$ have equal accuracy. To make this example closer to what we are familiar with, we use the example we used before; that is, we assume that y_n and $x^*_{n,n-1}$ represent two independent estimates of your weight obtained from two scales having equal accuracy (the example of Section 1.2.1). If one scale gives a weight estimate of 110 lb and the other 120 lb, what would you use for the best combined-weight estimate? You would take the average of the two weight estimates to obtain 115 lb. This is just what (2.3-1) does. If the variances of the two estimates are equal (say to σ^2), then (2.3-1) becomes

$$x^*_{n,n} = \left(\frac{x^*_{n,n-1}}{\sigma^2} + \frac{y_n}{\sigma^2}\right) \frac{1}{1/\sigma^2 + 1/\sigma^2} = \frac{x^*_{n,n-1} + y_n}{2} \tag{2.3-2}$$

Thus in Figure 1.2-3 the combined estimate $x^*_{n,n}$ is placed exactly in the middle between the two estimates y_n and $x^*_{n,n-1}$.

Now consider the case where $x^*_{n,n-1}$ is much more accurate than the estimate y_n. For this case $\mathrm{VAR}(x^*_{n,n-1}) \ll \mathrm{VAR}(y_n)$ or equivalently $1/\mathrm{VAR}(x^*_{n,n-1}) \gg 1/\mathrm{VAR}(y_n)$. As a result, (2.3-1) can be approximated by

$$\begin{aligned} x^*_{n,n} &= \left[\frac{x^*_{n,n-1}}{\mathrm{VAR}(x^*_{n,n-1})} + 0\right] \frac{1}{1/\mathrm{VAR}(x^*_{n,n-1}) + 0} \\ &\doteq x^*_{n,n-1} \end{aligned} \tag{2.3-3}$$

Thus the estimate $x^*_{n,n}$ is approximately equal to $x^*_{n,n-1}$, as it should be because the accuracy of $x^*_{n,n-1}$ is much better than that of y_n. For this case, in Figure 1.2-3 the combined estimate x^*_n is placed very close to the estimate $x^*_{n,n-1}$ (equal to it).

Equation (2.3-1) can be put in the form of one of the Kalman g–h tracking filters. Specifically, (2.3-1) can be rewritten as

$$x^*_{n,n} = x^*_{n,n-1} + \frac{\text{VAR}(x^*_{n,n})}{\text{VAR}(y_n)}(y_n - x^*_{n,n-1}) \tag{2.3-4}$$

This in turn can be rewritten as

$$x^*_{n,n} = x^*_{n,n-1} + g_n(y_n - x^*_{n,n-1}) \tag{2.3-5}$$

This is the same form as (1.2-7) [and also (1.2-8b)] for the g–h tracking filter. Comparing (2.3-5) with (1.2-7) gives us the expression for the constant g_n. Specifically

$$g_n = \frac{\text{VAR}(x^*_{n,n})}{\text{VAR}(y_n)} \tag{2.3-6}$$

Thus we have derived one of the Kalman tracking equations, the one for updating the target position. The equation for the tracking-filter parameter h_n is given by

$$h_n = \frac{\text{COV}(x^*_{n,n}\dot{x}^*_{n,n})}{\text{VAR}(y_n)} \tag{2.3-7}$$

A derivation for (2.3-7) is given for the more general case in Section 2.6.

2.4 KALMAN FILTER IN MATRIX NOTATION

In this section we shall rework the Kalman filter in matrix notation. The Kalman filter in matrix notation looks more impressive. You can impress your friends when you give it in matrix form! Actually there are very good reasons for putting it in matrix form. First, it is often put in matrix notation in the literature, and hence it is essential to know it in this form in order to recognize it. Second, and more importantly, as shall be shown later, in the matrix notation form the Kalman filter applies to a more general case than the one-dimensional case given by (2.1-3) or (1.2-11).

First we will put the system dynamics model given by (1.1-1) into matrix notation. Then we will put the random system dynamics model of (2.1-1) into matrix notation. Equation (1.1-1) in matrix notation is

$$X_{n+1} = \Phi X_n \tag{2.4-1}$$

where

$$X_n = \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} = \text{state vector} \tag{2.4-1a}$$

and

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$$
$$= \text{state transition matrix for constant-velocity trajectory [5, 43]} \tag{2.4-1b}$$

To show that (2.4-1) is identical to (1.1-1), we just substitute (2.4-1a) and (2.4-1b) into (2.4-1) to obtain

$$\begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} \tag{2.4-1c}$$

which on carrying out the matrix multiplication yields

$$\begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} = \begin{bmatrix} x_n + T\dot{x}_n \\ \dot{x}_n \end{bmatrix} \tag{2.4-1d}$$

which we see is identical to (1.1-1).

As indicated in (2.4-1a), X_n is the target trajectory state vector. This state vector is represented by a column matrix. As pointed out in Section 1.4, it consists of the quantities being tracked. For the filter under consideration these quantities are the target position and velocity at time n. It is called a two-state vector because it consists of two target states: target position and target velocity. Here, Φ is the state transition matrix. This matrix transitions the state vector X_n at time n to the state vector X_{n+1} at time $n+1$ a period T later.

It is now a simple matter to give the random system dynamics model represented by (2.1-1) in matrix form. Specifically, it becomes

$$X_{n+1} = \Phi X_n + U_n \tag{2.4-2}$$

where

$$U_n = \begin{bmatrix} 0 \\ u_n \end{bmatrix}$$
$$= \text{dynamic model driving noise vector} \tag{2.4-2a}$$

To show that (2.4-2) is identical to (2.1-1), we now substitute (2.4-1a), (2.4-1b),

and (2.4-2a) into (2.4-2) to obtain directly from (2.4-1d)

$$\begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} = \begin{bmatrix} x_n + T\dot{x}_n \\ \dot{x}_n \end{bmatrix} + \begin{bmatrix} 0 \\ u_n \end{bmatrix} \tag{2.4-2b}$$

which on adding the corresponding terms of the matrices on the right-hand side of (2.4-2b) yields

$$\begin{bmatrix} x_{n+1} \\ \dot{x}_{n+1} \end{bmatrix} = \begin{bmatrix} x_n + T\dot{x}_n \\ \dot{x}_n + u_n \end{bmatrix} \tag{2.4-2c}$$

which is identical to (2.1-1), as we desired to show.

We now put the trivial measurements equation given by (1.2-17) into matrix form. It is given by

$$Y_n = MX_n + N_n \tag{2.4-3}$$

where

$$M = [1 \quad 0] = \text{observation matrix} \tag{2.4-3a}$$

$$N_n = [\nu_n] = \text{observation error} \tag{2.4-3b}$$

$$Y_n = [y_n] = \text{measurement matrix} \tag{2.4-3c}$$

Equation (2.4-3) is called the observation system equation. This is because it relates the quantities being estimated to the parameter being observed, which, as pointed out in Section 1.5, are not necessarily the same. In this example, the parameters x_n and $\dot{x}_n$ (target range and velocity) are being estimated (tracked) while only target range is observed. In the way of another example, one could track a target in rectangular coordinates (x, y, z) and make measurements on the target in spherical coordinates (R, θ, ϕ). In this case the observation matrix M would transform from the rectangular coordinates being used by the tracking filter to the spherical coordinates in which the radar makes its measurements.

To show that (2.4-3) is given by (1.2-17), we substitute (2.4-3a) to (2.4-3c), into (2.4-3) to obtain

$$[y_n] = [1 \quad 0] \begin{bmatrix} x_n \\ \dot{x}_n \end{bmatrix} + [\nu_n] \tag{2.4-3d}$$

which on carrying out the multiplication becomes

$$[y_n] = [x_n] + [\nu_n] \tag{2.4-3e}$$

Finally, carrying out the addition yields

$$[y_n] = [x_n + \nu_n] \tag{2.4-3f}$$

which is identical to (1.2-17).

Rather than put the g–h tracking equations as given by (1.2-11) in matrix form, we will put (1.2-8) and (1.2-10) into matrix form. These were the equations that were combined to obtain (1.2-11). Putting (1.2-10) into matrix form yields

$$X^*_{n+1,n} = \Phi X^*_{n,n} \tag{2.4-4a}$$

where

$$X^*_{n,n} = \begin{bmatrix} x^*_{n,n} \\ \dot{x}^*_{n,n} \end{bmatrix} \tag{2.4-4b}$$

$$X^*_{n+1,n} = \begin{bmatrix} x^*_{n+1,n} \\ \dot{x}^*_{n+1,n} \end{bmatrix} \tag{2.4-4c}$$

This is called the prediction equation because it predicts the position and velocity of the target at time $n+1$ based on the position and velocity of the target at time n, the predicted position and velocity being given by the state vector of (2.4-4c). Putting (1.2-8) into matrix form yields

$$X^*_{n,n} = X^*_{n,n-1} + H_n(Y_n - MX^*_{n,n-1}) \tag{2.4-4d}$$

Equation (2.4-4d) is called the Kalman filtering equation because it provides the updated estimate of the present position and velocity of the target.

The matrix H_n is a matrix giving the tracking-filter constants g_n and h_n. It is given by

$$H_n = \begin{bmatrix} g_n \\ \dfrac{h_n}{T} \end{bmatrix} \tag{2.4-5}$$

for the two-state g–h or Kalman filter equations of (1.2-10). This form does not however tell us how to obtain g_n and h_n. The following form (which we shall derive shortly) does:

$$H_n = S^*_{n,n-1} M^T \left[R_n + MS^*_{n,n-1} M^T \right]^{-1} \tag{2.4-4e}$$

where

$$S^*_{n,n-1} = \Phi S^*_{n-1,n-1} \Phi^T + Q_n \quad \text{(predictor equation)} \tag{2.4-4f}$$

and

$$Q_n = \mathrm{COV}[U_n] = E[U_n U_n^T] \quad \text{(dynamic model noise covariance)} \tag{2.4-4g}$$

$$S^*_{n,n-1} = \mathrm{COV}(X^*_{n,n-1}) = E[X^*_{n,n-1} X^{*T}_{n,n-1}] \tag{2.4-4h}$$

$$R_n = \mathrm{COV}(N_n) = E[N_n N_n^T] \quad \text{(observation noise covariance)} \tag{2.4-4i}$$

$$\begin{aligned} S^*_{n-1,n-1} &= \mathrm{COV}(X^*_{n-1,n-1}) \\ &= [\mathrm{I} - H_{n-1}M] S_{n-1,n-2} \quad \text{(corrector equation)} \end{aligned} \tag{2.4-4j}$$

As was the case for (1.4-1), covariances in (2.4-4g) and (2.4-4i) apply as long as the entries of the column matrices U_n and N_n have zero mean. Otherwise U_n and N_n have to be replaced by $U_n - E[U_n]$ and $N_n - E[N_n]$, respectively. These equations at first look formidable, but as we shall see, they are not that bad. We shall go through them step by step.

Physically, the matrix $S^*_{n,n-1}$ is an estimate of our accuracy in prediciting the target position and velocity at time n based on the measurements made at time $n-1$ and before. Here, $S^*_{n,n-1}$ is the covariance matrix of the state vector $X^*_{n,n-1}$. To get a better feel for $S^*_{n,n-1}$, let us write it out for our two-state $X^*_{n,n-1}$. From (1.4-1) and (2.4-4c) it follows that

$$\begin{aligned} \mathrm{COV}\, X^*_{n,n-1} &= \overline{X^*_{n,n-1} X^{*T}_{n,n-1}} \\ &= \overline{\begin{bmatrix} x^*_{n,n-1} \\ \dot{x}^*_{n,n-1} \end{bmatrix} [x^*_{n,n-1} \quad \dot{x}^*_{n,n-1}]} = \begin{bmatrix} \overline{x^*_{n,n-1} x^*_{n,n-1}} & \overline{x^*_{n,n-1} \dot{x}^*_{n,n-1}} \\ \overline{\dot{x}^*_{n,n-1} x^*_{n,n-1}} & \overline{\dot{x}^*_{n,n-1} \dot{x}^*_{n,n-1}} \end{bmatrix} \\ &= \begin{bmatrix} \overline{x^{*2}_{n,n-1}} & \overline{x^*_{n,n-1} \dot{x}^*_{n,n-1}} \\ \overline{\dot{x}^*_{n,n-1} x^*_{n,n-1}} & \overline{\dot{x}^{*2}_{n,n-1}} \end{bmatrix} \\ &= \begin{bmatrix} s^*_{00} & s^*_{01} \\ s^*_{10} & s^*_{11} \end{bmatrix} = S^*_{n,n-1} \end{aligned} \tag{2.4-4k}$$

where for convenience $E[Z]$ has been replaced by $\bar{Z}$, that is, $E[\cdot]$ is replaced by the overbar. Again, the assumption is made that mean of $X^*_{n,n-1}$ has been substracted out in the above.

The matrix R_n gives the accuracy of the radar measurements. It is the covariance matrix of the measurement error matrix N_n given by (2.4-4i). For our two-state filter with the measurement equation given by (2.4-3) to (2.4-3c),

$$\begin{aligned} R_n &= \mathrm{COV}\,[N_n] = \overline{[\nu_n][\nu_n]}^T = \overline{[\nu_n][\nu_n]} \\ &= \overline{[\nu_n^2]} = \overline{[\nu_n^2]} \\ &= [\sigma_\nu^2] = [\sigma_x^2] \end{aligned} \tag{2.4-4l}$$

where it is assumed as in Section 1.2.4.4 that σ_ν and σ_x are the rms of ν_n independent of n. Thus σ_ν^2 and σ_x^2 are the variance of ν_n, the assumption being that the mean of ν_n is zero; see (1.2-18).

The matrix Q_n, which gives the magnitude of the target trajectory uncertainty or the equivalent maneuvering capability, is the covariance matrix of the dynamic model driving noise vector, that is, the random-velocity component of the target trajectory given by (2.4-2a); see also (2.1-1). To get a better feel for Q_n, let us evaluate it for our two-state Kalman filter, that is, for U_n given by (2.4-2a). Here

$$Q_n = \mathrm{COV}U_n = \overline{U_n U_n^T} = \overline{\begin{bmatrix} 0 \\ u_n \end{bmatrix} [0 \quad u_n]}$$
$$= \overline{\begin{bmatrix} 0 \cdot 0 & 0 \cdot u_n \\ u_n \cdot 0 & u_n \cdot u_n \end{bmatrix}} = \begin{bmatrix} 0 & 0 \\ 0 & \overline{u_n^2} \end{bmatrix} \tag{2.4-4m}$$

Equation (2.4-4f) allows us to obtain the prediction covariance matrix $S^*_{n,n-1}$ from the covariance matrix of the filtered estimate of the target state vector at

TABLE 2.4-1. Kalman Equation

Predictor equation:

$$X^*_{n+1,n} = \Phi X^*_{n,n} \tag{2.4-4a}$$

Filtering equation:

$$X^*_{n,n} = X^*_{n,n-1} + H_n(Y_n - MX^*_{n,n-1}) \tag{2.4-4d}$$

Weight equation:

$$H_n = S^*_{n,n-1} M^T [R_n + MS^*_{n,n-1} M^T]^{-1} \tag{2.4-4e}$$

Predictor covariance matrix equation:

$$S^*_{n,n-1} = \mathrm{COV}(X^*_{n,n-1}) \tag{2.4-4h}$$
$$S^*_{n,n-1} = \Phi S^*_{n-1,n-1} \Phi^T + Q_n \tag{2.4-4f}$$

Covariance of random system dynamics model noise vector U^a:

$$Q_n = \mathrm{COV}(U_n) = E[U_n U_n^T] \tag{2.4-4g}$$

Covariance of measurement vector $Y_n = X_n + N_n^a$:

$$R_n = \mathrm{COV}(Y_n) = \mathrm{COV}(N_n) = E[N_n N_n^T] \tag{2.4-4i}$$

Corrector equation (covariance of smoothed estimate):

$$S^*_{n-1,n-1} = \mathrm{COV}(X^*_{n-1,n-1}) = (I - H_{n-1}M) S^*_{n-1,n-2} \tag{2.4-4j}$$

[a] If $E[U] = E[N_n] = 0$.

time $n-1$ given by $S^*_{n-1,n-1}$. The filtered estimate covariance matrix $S^*_{n-1,n-1}$ is in turn obtained from the previous prediction covariance matrix $S^*_{n-1,n-2}$ using (2.4-4j). Equations (2.4-4e), (2.4-4f), and (2.4-4j) allow us to obtain the filter weights H_n at successive observation intervals. For the two-state *g–h* filter discussed earlier, the observation matrix is given by (2.4-3a) and the filter coefficient matrix H_n is given by (2.4-5). The covariance matrix for the initial a priori estimates of the target position and velocity given by $S^*_{0,-1}$ allows initiation of the tracking equations given by (2.4-4d). First (2.4-4e) is used to calculate H_0 (assuming that $n=0$ is the time for the first filter observation). For convenience the above Kalman filter equations are summarized in Table 2.4-1.

The beauty of the matrix form of the Kalman tracking-filter equations as given by (2.4-4) is, although presented here for our one-dimensional (range only), two-state (position and velocity) case, that the matrix form applies in general. That is, it applies for tracking in any number of dimensions for the measurement and state space and for general dynamics models. All that is necessary is the proper specification of the state vector, observation matrix, transition matrix, dynamics model, and measurement covariance matrix. For example, the equations apply when one is tracking a ballistic target in the atmosphere in three dimensions using rectangular coordinates (x, y, z) with a ten-state vector given by

$$X^*_{n,n-1} = \begin{bmatrix} x^*_{n,n-1} \\ \dot{x}^*_{n,n-1} \\ \ddot{x}^*_{n,n-1} \\ y^*_{n,n-1} \\ \dot{y}^*_{n,n-1} \\ \ddot{y}^*_{n,n-1} \\ z^*_{n,n-1} \\ \dot{z}^*_{n,n-1} \\ \ddot{z}^*_{n,n-1} \\ \beta^*_{n,n-1} \end{bmatrix} \tag{2.4-6}$$

where β is the atmospheric drag on the target. One can assume that the sensor measures R, θ, ϕ, and the target Doppler $\dot{R}$ so that Y_n is given by

$$Y_n = \begin{bmatrix} R_n \\ \dot{R}_n \\ \theta_n \\ \phi_n \end{bmatrix} \tag{2.4-7}$$

In general the vector Y_n would be given by

$$Y_n = \begin{bmatrix} y_{1n} \\ y_{2n} \\ \vdots \\ y_{mn} \end{bmatrix} \tag{2.4-8}$$

where y_{in} is the ith target parameter measured by the sensor at time n.

The atmosheric ballistic coefficient β is given by

$$\beta = \frac{m}{C_D A} \tag{2.4-9}$$

where m is the target mass, C_D is the atmospheric dimensionless drag coefficient dependent on the body shape, and A is the cross-sectional area of the target perpendicular to the direction of motion. [See (16.3-18), (16.3-19), (16.3-27) and (16.3-28) of Section 16.3 for the relation between drag constant and target atmospheric deceleration.]

For the g–h Kalman filter whose dynamics model is given by (2.1-1) or (2.4-2), the matrix Q is given by (2.4-4m), which becomes

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_u^2 \end{bmatrix} \tag{2.4-10}$$

if it is assumed that the mean of u_n is zero and its variance is σ_u^2 independent of n. For the equivalent g–h–k Kalman filter to our two-state g–h Kalman filter having the dynamic model of (2.4-2), the three-state dynamics model is given by (1.3-3) with (1.3-3a) replaced by

$$\ddot{x}^*_{n+1,n} = \ddot{x}^*_{n,n} + w_n \tag{2.4-11}$$

where w_n equals a random change in acceleration from time n to $n+1$. We assume w_n is independent from n to $n+1$ for all n and that it has a variance σ_w^2. Physically w_n represents a random-acceleration jump occurring just prior to the $n+1$ observation. For this case

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_w^2 \end{bmatrix} \tag{2.4-12}$$

The variance of the target acceleration dynamics σ_w^2 (also called σ_a^2) can be specified using an equation similar to that used for specifying the target velocity dynamics for the Kalman g–h filter. Specifically

$$\sigma_w = \frac{T\ddot{x}_{\max}}{C} \tag{2.4-13}$$

where C is a constant and $\ddot{x}_{\max}$ is the maximum $\ddot{x}$. For the steady-state g–h–k Kalman filter for which Q is given by (2.4-12) g,h, and k are related by (1.3-10a) to (1.3-10c) [11, 14, 15] and σ_a^2, σ_x^2, and T are related to g and k by [14]

$$\frac{T^4\sigma_a^2}{4\sigma_x^2} = \frac{k^2}{1-g} \tag{2.4-14}$$

For the general g–h–k Kalman filter (2.4-5) becomes [14]

$$H_n = \begin{bmatrix} g_n \\ \dfrac{h_n}{T} \\ \dfrac{2k_n}{T^2} \end{bmatrix} \tag{2.4-15}$$

This is a slightly underdamped filter, just as is the steady-state g–h Kalman filter that is the Benedict–Bordner filter. Its total error $E_{TN} = 3\sigma_{n+1,n} + b^*$ is less than that for the critically damped g–h–k filter, and its transient response is about as good as that of the critical damped filter [11]. In the literature, this steady-state Kalman filter has been called the optimum g–h–k filter [11].

If we set $\sigma_u^2 = 0$ in (2.4-10), that is, remove the random maneuvering part of the Kalman dynamics, then

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.4-16}$$

and we get the growing-memory filter of Section 1.2.10, the filter used for track initiation of the constant g–h filters.

2.5 DERIVATION OF MINIMUM-VARIANCE EQUATION

In Section 2.3 we used the minimum-variance equation (2.3-1) to derive the two-state Kalman filter range-filtering equation. We will now give two derivations of the minimum-variance equation.

2.5.1 First Derivation

The first derivation parallels that of reference 7. For simplicity, designate the two independent estimates $x_{n,n-1}^*$ and y_n by respectively x_1^* and x_2^*. Designate $x_{n,n}^*$, the optimum combined estimate, by x_c^*. We desire to find an optimum linear estimate for x_c^*. We can designate this linear estimate as

$$x_c^* = k_1 x_1^* + k_2 x_2^* \tag{2.5-1}$$

We want this estimate x_c^* to be unbiased, it being assumed that x_1^* and x_2^* are unbiased. Designate as x the true value of x. Obtaining the mean of (2.5-1), it follows that for the estimate to be unbiased

$$x = k_1 x + k_2 x \tag{2.5-2}$$

which becomes

$$1 = k_1 + k_2 \tag{2.5-3}$$

Thus for the estimate to be unbiased we require

$$k_2 = 1 - k_1 \tag{2.5-4}$$

Substituting (2.5-4) into (2.5-1) yields

$$x_c^* = k_1 x_1^* + (1 - k_1) x_2^* \tag{2.5-5}$$

Let the variances of x_c^*, x_1^*, and x_2^* be designated as respectively σ_c^2, σ_1^2, and σ_2^2. Then (2.5-5) yields

$$\sigma_c^2 = k_1^2 \sigma_1^2 + (1 - k_1)^2 \sigma_2^2 \tag{2.5-6}$$

To find the k_1 that gives the minimum σ_c^2, we differentiate (2.5-6) with respect to k_1 and set the result to zero, obtaining

$$2 k_1 \sigma_1^2 - 2(1 - k_1)\sigma_2^2 = 0 \tag{2.5-7}$$

Hence

$$k_1 = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{2.5-8}$$

Substituting (2.5-8) into (2.5-5) yields

$$x_c^* = \frac{\sigma_2^2}{\sigma_1^2 + \sigma_2^2} x_1^* + \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} x_2^* \tag{2.5-9}$$

Rewriting (2.5-9) yields

$$x_c^* = \left(\frac{x_1^*}{\sigma_1^2} + \frac{x_2^*}{\sigma_2^2} \right) \frac{1}{1/\sigma_1^2 + 1/\sigma_2^2} \tag{2.5-10}$$

which is identical to Eq. (2.3-1), as we desired to show. Note that substituting

(2.5-8) into (2.5-6) yields

$$\sigma_c^2 = \left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}\right)^{-1} \tag{2.5-11}$$

2.5.2 Second Derivation

The second derivation employs a weighted least-squares error estimate approach. In Figure 1.2-3 we have two estimates y_n and $x^*_{n,n-1}$ and desire here to replace these with a combined estimate $x^*_{n,n}$ that has a minimum weighted least-squares error. For an arbitrarily chosen $x^*_{n,n}$ shown in Figure 1.2-3 there are two errors. One is the distance of $x^*_{n,n}$ from y_n; the other is its distance of $x^*_{n,n}$ from $x^*_{n,n-1}$. For the minimum least-squares estimate in Section 1.2.6 we minimized the sum of the squares of the distances (errors) between the measurements and the best-fitting line (trajectory) to the measurements. We would like to similarly minimize the sum of the two errors here between $x^*_{n,n}$ and y_n and $x^*_{n,n-1}$ in some sense. One could minimize the sum of the squares of the errors as done in Section 1.2.6, but this is not the best tactic because the two errors are not always equally important. One of the estimates, either y_n or $x^*_{n,n-1}$, will typically be more accurate than the other. For convenience let us say $x^*_{n,n-1}$ is more accurate than y_n. In this case it is more important that $(x^*_{n,n-1} - x^*_{n,n})^2$ be small, specifically smaller than $(y_n - x^*_{n,n})^2$. This would be achieved if in finding the least sum of the squares of each of the two errors we weighted the former error by a larger constant than the latter error. We are thus obtaining a minimization of an appropriately weighted sum of the two errors wherein the former receives a larger weighting. A logical weighting is to weight each term by 1 over the accuracy of their respective estimates as the following equation does:

$$E = \frac{(y_n - x^*_{n,n})^2}{\text{VAR}y_n} + \frac{(x^*_{n,n-1} - x^*_{n,n})^2}{\text{VAR}(x^*_{n,n-1})} \tag{2.5-12}$$

Here the error $(x^*_{n,n-1} - x^*_{n,n})^2$ is weighted by 1 over the variance of $x^*_{n,n-1}$ and $(y_n - x^*_{n,n})^2$ by 1 over the variance of y_n. Thus if $\text{VAR}(x^*_{n,n-1}) \ll \text{VAR}(y_n)$, then $1/\text{VAR}(x^*_{n,n-1}) \gg 1/\text{VAR}(y_n)$ and forces the error $(x^*_{n,n-1} - x^*_{n,n})^2$ to be much smaller than the error $(y_n - x^*_{n,n})^2$ when minimizing the weighted sum of errors E of (2.5-12). This thus forces $x^*_{n,n}$ to be close to $x^*_{n,n-1}$, as it should be. The more accurate $x^*_{n,n-1}$, the closer $x^*_{n,n}$ is to $x^*_{n,n-1}$. In (2.5-12) the two errors are automatically weighted according to their importance, the errors being divided by their respective variances. On finding the $x^*_{n,n}$ that minimizes E of (2.5-12), a weighted least-squares estimate instead of just a least-squares estimate is obtained. This is in contrast to the simple unweighted least-squares estimate obtained in Section 1.2.6.

It now remains to obtain the $x^*_{n,n}$ that minimizes (2.5-12). This is a straightforward matter. Differentiating (2.5-12) with respect to $x^*_{n,n}$ and setting the result equal to zero yields

$$\frac{dE}{dx^*_{n,n}} = \frac{2(y_n - x^*_{n,n})}{\text{VAR}(y_n)} + \frac{2(x^*_{nn-1} - x^*_{n,n})}{\text{VAR}(x^*_{n,n-1})} = 0 \qquad (2.5\text{-}13)$$

Solving for $x^*_{n,n}$ yields (2.3-1), the desired result.

2.6 EXACT DERIVATION OF *r*-DIMENSIONAL KALMAN FILTER

We will now extend the second derivation given in Section 2.5.2 to the case where a target is tracked in *r*-dimensions. An example of an *r*-dimensional state vector for which case $r = 10$ is given by (2.4-6). For this case the target is tracked in the three-dimensional rectangular (x, y, z) coordinates in position, velocity, and acceleration. In addition, the atmospheric drag parameter β is also kept track of to form the 10th parameter of the 10-dimensional state vector $X^*_{n,n-1}$.

The 10 states of the state vector are to be estimated. The measurements made on the target at time n are given by the measurement matrix Y_n. As indicated in Section 1.5, the measurements made on the target need not be made in the same coordinate system as the coordinate system used for the state vector $X^*_{n,n-1}$. For example the target measurements are often made in a spherical coordinate system consisting of slant range R, target azimuth angle θ, and target elevation angle ϕ, yet the target could be tracked in rectangular coordinates. If the target Doppler velocity $\dot{R}$ is measured, then Y_n becomes (2.4-7). The observation matrix M of (2.4-3) converts the predicted trajectory state vector $X^*_{n,n-1}$ from its coordinate system to the coordinate system used for making the radar measurements, that is, the coordinate system of Y_n.

For simplicity let us assume initially that the coordinates for the *r*-dimensional state vector $X^*_{n,n-1}$ is the same as for Y_n. Let $X^*_{n,n}$ be our desired combined estimate of the state vector after the measurement Y_n. The combined estimate $X^*_{n,n}$ will lie somewhere in between the predicted state vector $X^*_{n,n-1}$ and the measurement vector Y_n as was the case in the one-dimensional situation depicted in Figure 1.2-3. Figure 2.6-1 shows the situation for our present multidimensional case. As done in the second derivation for the one-dimensional case discussed in Section 2.5.2, we will choose for our best combined estimate the $X^*_{n,n}$ that minimizes the weighted sum of the error differences between Y_n and $X^*_{n,n}$ and between $X^*_{n,n-1}$ and $X^*_{n,n}$. Again the weighting of these errors will be made according to their importance. The more important an error is, the smaller it will be made. An error is deemed to be more important if it is based on a more accurate estimate of the position of the target.

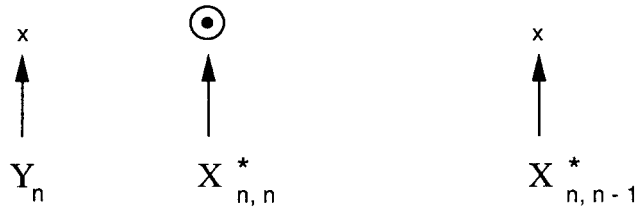

Figure 2.6-1 Filtering problem. Determination of $X^*_{n,n}$ based on measurement Y_n and prediction $X^*_{n,n-1}$.

Accordingly, as done for the one-dimensional case, we weight the square of the errors by 1 over the variance of the error. Thus the equivalent equation to (2.5-12) becomes

$$J = \frac{(Y_n - X^*_{n,n})^2}{\mathrm{VAR}\, Y_n} + \frac{(X^*_{n,n-1} - X^*_{n,n})^2}{\mathrm{VAR} X^*_{n,n-1}} \tag{2.6-1}$$

This equation is only conceptually correct; the mathematically correct equivalent will be given shortly.

It remains now to use (2.6-1) to solve for the new combined estimate $X^*_{n,n}$ that minimizes the weighted sum of the squares of the errors. Conceptually, this is done just as it was done for the equivalent one-dimensional (2.5-12). Specifically, (2.6-1) is differentiated with respect to the combined estimate $X^*_{n,n}$ with the resulting equation set equal to zero in order to solve for the combined estimate $X^*_{n,n}$ that minimizes the weighted sum of the errors squared. There is one problem though: (2.6-1) is not correct when one is dealing with matrices. It is only conceptually correct as indicated.

When using matrix notation, the first term on the right of (2.6-1) must be written as

$$\frac{(Y_n - X^*_{n,n})^2}{\mathrm{VAR}\, Y_n} \equiv (Y_n - X^*_{n,n})^T R_n^{-1} (Y_n - X^*_{n,n}) \tag{2.6-2}$$

Where the matrix R_n is the covariance matrix Y_n, that is,

$$R_n = \mathrm{COV}(Y_n) \tag{2.6-3}$$

which is the same as defined by (2.4-4i). The inverse of the covariance matrix R_n, which is designed as R_n^{-1}, takes the place of dividing by the variance of Y_n when dealing with matrices. Note that if R_n is diagonal with all the diagonal

terms equal to σ_x^2, then (2.6-2) becomes

$$\frac{(Y_n - X_{n,n}^*)^2}{\text{VAR}\, Y_n} \equiv \frac{(Y_n - X_{n,n}^*)^T (Y_n - X_{n,n}^*)}{\sigma_x^2} = \frac{\sum_{i=1}^{r} \left((y_{in} - x_{i,nn}^* \right)^2}{\sigma_x^2} \tag{2.6-4}$$

If the coordinate system for Y_n and $X_{n,n}^*$ are not the same, then (2.6-2) becomes

$$\frac{(Y_n - X_{n,n}^*)^2}{\text{VAR}\, Y_n} \equiv \left(Y_n - MX_{n,n}^{*T} \right) R_n^{-1} (Y_n - MX_{n,n}^*) \tag{2.6-5}$$

The corresponding correct form for the second term on the right of (2.6-1) is

$$\frac{(X_{n,n-1}^* - X_{n,n}^*)^2}{\text{VAR} X_{n,n-1}^*} \equiv (X_{n,n-1}^* - X_{n,n}^*)^T S_{n,n-1}^{*-1} (X_{n,n-1}^* - X_{n,n}^*) \tag{2.6-6}$$

where $S_{n,n-1}^*$ is the covariance matrix of $X_{n,n-1}^*$; see (2.4-4h) and (2.4-4f). Substituting (2.6-5) and (2.6-6) into (2.6-1) yields

$$J = (Y_n - MX_{n,n}^*)^T R_n^{-1} (Y_n - MX_{n,n}^*) + (X_{n,n-1}^* - X_{n,n}^*)^T S_{n,n-1}^{*-1} (X_{n,n-1}^* - X_{n,n}^*) \tag{2.6-7}$$

Now we are in a position to solve for $X_{n,n}^*$. As discussed before, this is done by differentiating (2.6-7) with respect to $X_{n,n}^*$, setting the resulting equation equal to zero in solving for $X_{n,n}^*$. The details of this are carried out in the remaining paragraphs of this section. The results are the full-blown Kalman filter equations given by (2.4-4) and summarized in Table 2.4-1. The reader may forgo the detailed mathematical derivation that follows in the next few paragraphs. However, the derivation is relatively simple and straight forward. At some point it is recommended that the reader at least glance at it, the Kalman filter having had such a major impact on filtering theory and the derivation given here being of the simplest of the full-blown Kalman filter that this author has seen.[†] The derivation makes use of matrix differentiation. The reader not familiar with matrix differentiation will be able to learn it by following the steps of the derivation given. Matrix differentiation is really very simple, paralleling standard algebraic differentiation in which

$$\frac{d(x^2)}{dx} = 2x \tag{2.6-8}$$

[†] This derivation was pointed out to the author by Fred Daum of the Raytheon Company.

Differentiation of a matrix equation, such as that of (2.6-7), is achieved by obtaining the gradient of the matrix equation as given by

$$\text{Gradient of } J \triangleq \frac{\partial J}{\partial X^*_{n,n}} = \left[\frac{\partial J}{\partial x_1}, \frac{\partial J}{\partial x_2}, \cdots, \frac{\partial J}{\partial x_n}\right] \tag{2.6-9}$$

Applying (2.6-9) to (2.6-7) yields

$$\frac{\partial J}{\partial X} = 2(X^*_{n,n} - X^*_{n,n-1})^T S^{*-1}_{n,n-1} + 2(Y_n - MX^*_{n,n})^T R_n^{-1}(-M) = 0 \tag{2.6-10}$$

This can be rewritten as

$$X^{*T}_{n,n}\left(S^{*-1}_{n,n} + M^T R_n^{-1} M\right) = X^{*T}_{n,n-1} S^{*-1}_{n,n-1} + Y_n^T R_n^{-1} M \tag{2.6-11}$$

which on taking the transpose of both sides and using $(AB)^T = B^T A^T$ yields

$$\left(S^{*-1}_{n,n} + M^T R_n^{-1} M\right) X^*_{n,n} = S^{*-1}_{n,n-1} X^*_{n,n-1} + M^T R_n^{-1} Y_n \tag{2.6-12}$$

or

$$X^*_{n,n} = \left(S^{*-1}_{n,n-1} + M^T R_n^{-1} M\right)^{-1} \left(S^{*-1}_{n,n-1} X^*_{n,n-1} + M^T R_n^{-1} Y_n\right) \tag{2.6-13}$$

The well-known matrix inversion lemma [5] states

$$\left(S^{*-1}_{n,n} + M^T R_n^{-1} M\right)^{-1} = S^*_{n,n-1} - S^*_{n,n-1} M^T (R_n + M S^*_{n,n-1} M^T)^{-1} M S^*_{n,n-1} \tag{2.6-14}$$

This can be rewritten as

$$\left(S^{*-1}_{n,n} + M^T R_n^{-1} M\right)^{-1} = S^*_{n,n-1} - H_n M S^*_{n,n-1} \tag{2.6-15}$$

where, as given by (2.4-4e),

$$H_n = S^*_{n,n-1} M^T (R_n + M S^*_{n,n-1} M^T)^{-1} \tag{2.6-15a}$$

Substituting (2.6-15) into (2.6-13) yields

$$\begin{aligned} X^*_{n,n} &= \left[S^*_{n,n-1} - H_n M S^*_{n,n-1}\right]\left[S^{*-1}_{n,n-1} X^*_{n,n-1} + M^T R^{-1} Y_n\right] \\ &= X^*_{n,n-1} - H_n M X^*_{n,n-1} + (S^*_{n,n-1} - H_n M S^*_{n,n-1}) M^T R^{-1} Y_n \end{aligned} \tag{2.6-16}$$

But as shall be shown shortly,

$$H_n = (S^*_{n,n-1} - H_n M S^*_{n,n-1}) M^T R^{-1} \tag{2.6-17}$$

Hence (2.6-16) can be written as

$$X^*_{n,n} = X^*_{n,n-1} + H_n Y_n - H_n M X^*_{n,n-1} \tag{2.6-18}$$

or

$$X^*_{n,n} = X^*_{n,n-1} + H_n (Y_n - M X^*_{n,n-1}) \tag{2.6-19}$$

But (2.6-19) is identical to the Kalman filter equation given by (2.4-4d), which is what we set out to prove.

We will now prove (2.6-17). From (2.6-15a) it follows that

$$S^*_{n,n-1} M^T = H_n (R_n + M S^*_{n,n-1} M^T) \tag{2.6-20}$$

This equation can be rewritten as

$$S^*_{n,n-1} M^T R_n^{-1} - H_n M S^*_{n,n-1} M^T R_n^{-1} = H_n \tag{2.6-21}$$

which in turn becomes (2.6-17), as we set out to derive.

The corrector equation (2.4-4j) follows from (2.6-19) and (2.6-21). The predictor equation (2.4-4f) follows from (2.4-2) and (2.4-4a). This completes out derivation of the Kalman equations given by (2.4-4a) through (2.4-4j).

2.7 TABLE LOOKUP APPROXIMATION TO THE KALMAN FILTER

An approximation to the Kalman filter can be used that involves a table lookup instead of the use of (2.4-4e) to calculate the coefficients of the matrix H_n in (2.4-4d). One such approximate lookup table is given in Table 2.7-1. As indicated in the table the coefficients g and h are determined by the sequence of detection hits and misses observed for the target in track. Also given in this table is the size of the search window to be used for a given track update. The approximate lookup procedure given in Table 2.7-1 is similar to that used for the ARTS III filter [21].

2.8 ASQUITH–FRIEDLAND STEADY-STATE *g–h* KALMAN FILTER

It was indicated earlier that the steady-state Kalman filter for the target dynamics model given by (2.1-1) yields the Benedict–Bordner *g–h* filter for

TABLE 2.7-1. Table Lookup Approximation to Kalman Filter

	Tracking Parameter Lookup		
Firmness, F_n	Position Smooth, g_n	Velocity Smooth, h_n	Search Bin, Δ_n
0	1.000	.000	—
1	1.000	1.000	$\Delta t \cdot \mathrm{v}_{\max}$
2	.833	.700	21.0σ
3	.700	.409	11.7σ
4	.600	.270	8.8σ
⋮	⋮	⋮	⋮
15	.228	.030	4.5σ

Note: If "hit" at n: $F_{n+1} = F_n + 1\,(F_{n,\max} = 15)$. If "miss": $F_{n+1} = F_n - 2\,(F_{n,\min} = 1)$. Initial hit: $F_0 = 0 \rightarrow F_1 = 1$. These are similar but not identical to ARTS III filter.

Source: After Sittler [21].

which g and h are related by (1.2-27). An alternate target dynamics model for the two-state Kalman filter is given by [22, 23]

$$x_{n+1} = x_n + \dot{x}_n T + \tfrac{1}{2} a_n T^2 \tag{2.8-1a}$$

$$\dot{x}_{n+1} = \dot{x}_n + a_n T \tag{2.8-1b}$$

where a_n is a random acceleration occurring between time n and $n+1$. The random acceleration a_n has the autocorrelation function given by

$$\overline{a_n a_m} = \begin{cases} \sigma_a^2 & \text{for } n = m \\ 0 & \text{for } n \neq m \end{cases} \tag{2.8-2}$$

Hence a_n is characterized as white noise. This model differs from that of (2.1-1) in that here a_n is a random acceleration that is constant between the time n and $n+1$ measurements whereas in (2.9-1) a random-velocity jump occurs just before the $(n+1)$st observation.

Figure 2.8-1 compares the possible g, h pairings for the Asquith–Friedland filter [(2.8-4)] obtained using the dynamics model of (2.8-1) and (2.8-2) with those of the Benedict–Bordner filter [(2.1-4)] having the dynamic model given by (2.1-1) or (2.4-2) with COV(U_n) given by (2.4-10). Also shown for comparison are the g and h weight pairings for the critically damped filter [1.2-36)] and the expanding-memory polynomial filter described in Sections 1.2.6 and 1.2.10, respectively. Recall that the expanding-memory polynomial filter [(1.2-38)] is a Kalman filter for which there is no noise term in the target dynamics model, that is, $u_n = 0$ in (2.1-1b); see (2.4-16) for the resulting Q for this target model. Note that the steady-state Asquith–Friedland filter (referred to as the discrete Kalman filter in reference 22) has viable designs for $h > 1$;

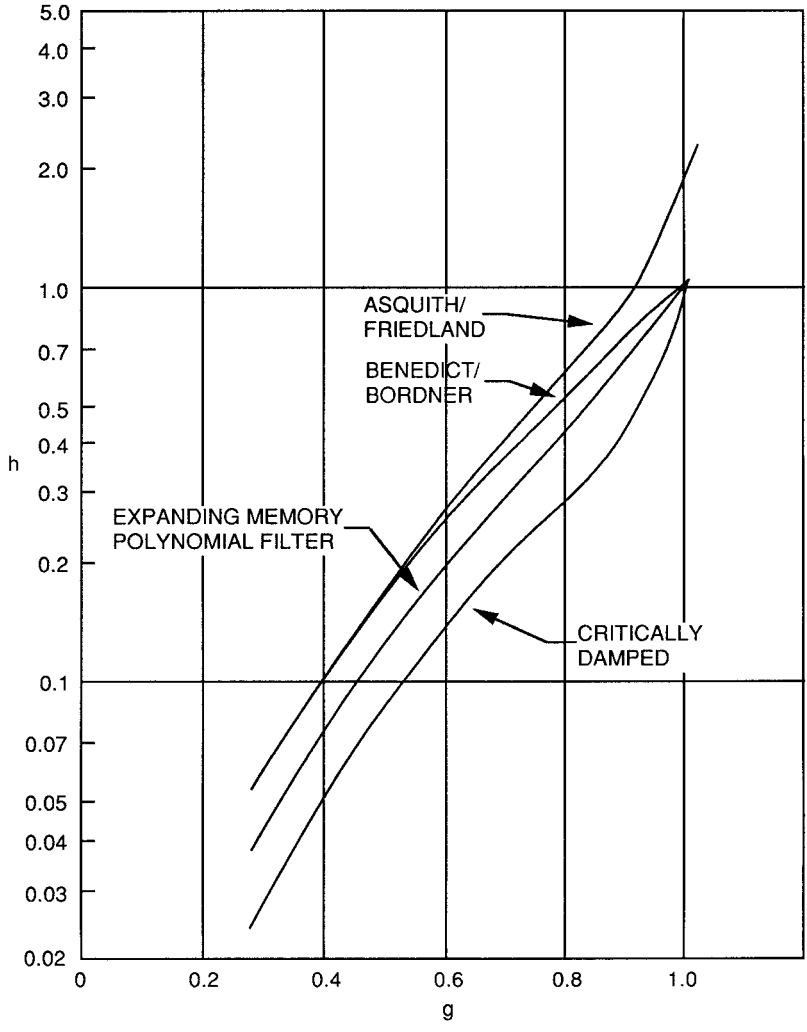

Figure 2.8-1 Comparison of g and h parameters for several g–h filters. (After Asquith [22].)

specifically it can have $h = 1.8755$ and $g = 0.9990$. This pair of values still leads to a stable g–h filter; that is, it falls in the triangular region of Figure 1.2-17 where g–h filters are stable. In fact, all the curves of Figure 2.8-1 fall within the triangular region. The figure indicates that the Asquith–Friedland and Benedict–Bordner filter are approximately the same for $g < 0.5$. Of these four constant g–h filters, the Asquith–Friedland filter is the most underdamped followed by the Benedict–Bordner filter, the expanding-memory filter, and the critically damped filter. Figure 2.8-2 plots a normalized dynamic error $b^* = b^*_{n+1,n}$ for three of these g–h filters versus the rms predicted position error.

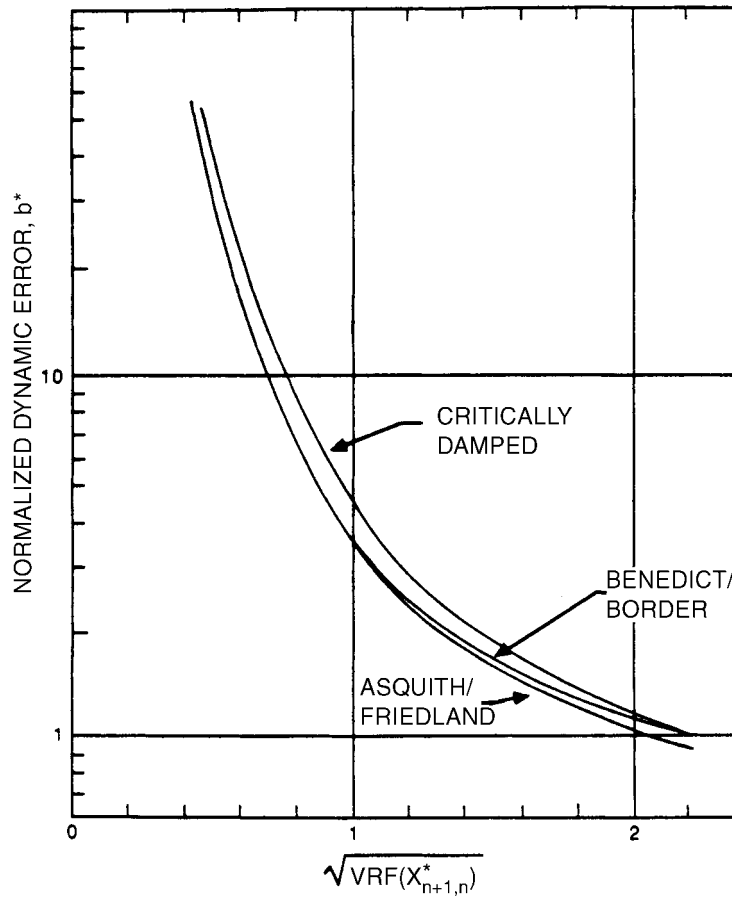

Figure 2.8-2 Normalized dynamic error $b^*/T^2\ddot{x} = 1/h$ versus square root of one-step prediction variance reduction factor for several g–h filters. (Asquith [22].)

It is easy to show that the Q matrix of (2.4-4g) for a target having the dynamic model given by (2.8-1a) and (2.8-1b) is given by [22]

$$Q = T^2 \begin{bmatrix} \sigma_a^2 \dfrac{T^2}{4} & \sigma_a^2 \dfrac{T}{2} \\ \sigma_a^2 \dfrac{T}{2} & \sigma_a^2 \end{bmatrix} \tag{2.8-3}$$

In steady state [22]

$$h = 4 - 2g - 4\sqrt{1-g} \tag{2.8-4}$$

and

$$T^2 \frac{\sigma_a^2}{\sigma_x^2} = \frac{h^2}{1-g} \tag{2.8-5}$$

which note is similar to (2.1-5) obtained for Q given by (2.4-10).

2.9 SINGER *g–h–k* KALMAN FILTER

In this section and the next we will give some feel for the Singer *g–h–k* Kalman filter [6, 8, 24], indicating the type of maneuvering target for which it is designed and then give some performance results.

For this filter Singer specified a target dynamics model for which the acceleration is a random function of time whose autocorrelation function is given by

$$E[\ddot{x}(t)\ddot{x}(t+t')] = \sigma_a^2 \exp\left(-\frac{|t'|}{\tau}\right) \tag{2.9-1}$$

where τ is the correlation time of the acceleration that could be due to a target maneuver, a target turn, or atmospheric turbulence. For a lazy turn τ is typically up to 60 sec, for an evasive maneuver τ is typically between 10 and 30 sec, while atmospheric turbulence results in a correlation time of the order of 1 or 2 sec. It is further assumed by Singer that the target acceleration has the probability density function given by Figure 2.9-1. This figure indicates that the target can have a maximum acceleration of $\pm A_{\max}$ with probability $P_{\max}$, no acceleration with probability P_0, and an acceleration between $\pm A_{\max}$ with a probability given by the uniform density function amplitude given in Figure 2.9-1.

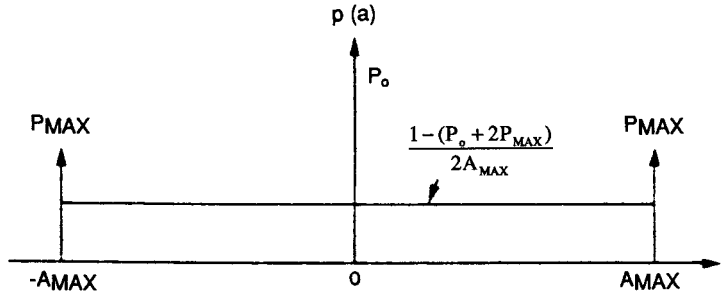

Figure 2.9-1 Model used by Singer [24] for target acceleration probability density function. (After Singer, R. A. "Estimating Optimal Tracking Filter Performance for Manned Maneuvering Targets," IEEE Trans. on Aerospace and Electronic Systems, Vol. AES-6(4), 1970. © 1970, IEEE.)

The total variance of the acceleration is given by

$$\sigma_a^2 = \frac{A_{\max}^2}{3}(1 + 4P_{\max} - P_0) \tag{2.9-2}$$

To apply the Kalman filter as given by (2.4-4), we need a white noise for the target model velocity jump term; that is, the velocity jump u_n of (2.4-2a) must be independent from time n to time $n+1$. For the target dynamics as given by (2.9-1) we find that the velocity jump u_n of (2.4-2a) is correlated. Even though the actual acceleration term is correlated, a white-noise acceleration forcing term can be generated in its place. This is done by finding the circuit that when driven by white noise $n_a(\tau)$ gives the correlated acceleration $\ddot{x}(\tau)$ as its output with the autocorrelation function for the output given by (2.9-1).

The transfer function for the filter that achieves this is given by

$$H(\omega) = \frac{\tau}{1 + j\omega\tau} \tag{2.9-3}$$

where $\omega = 2\pi f$. The inverse of the above filter $H_a(\omega)$ is the Wiener–Kolmogorov whitening filter. The differential equation for this filter is given by

$$\frac{d}{dt}(\ddot{x}) = -\frac{1}{\tau}\ddot{x} + n_a(t) \tag{2.9-4}$$

To use the driving term with the white-noise acceleration $n_a(\tau)$ in place of the autocorrelated acceleration $\ddot{x}_n$ requires the augmentation of the number of states in the tracking filter from 2 to 3. Thus

$$X_n = \begin{bmatrix} x_n \\ \dot{x}_n \\ \ddot{x}_n \end{bmatrix} \tag{2.9-5}$$

instead of $X_n = [x_n \quad \dot{x}_n]^T$. Consequently, a *g–h–k* filter results instead of a *g–h* filter.

We shall show shortly that for the Singer dynamics model the three-state dynamics equation equivalent to (2.1-1) then takes the form

$$X_{n+1} = \Phi X_n + U_n \tag{2.9-6}$$

when the whitening filter of (2.9-4) is used, which allows us to replace the correlated acceleration with white-noise acceleration. The three-state dynamics driving noise U_n is now

$$U_n = \begin{bmatrix} u_{1,n} \\ u_{2,n} \\ u_{3,n} \end{bmatrix} \tag{2.9-7}$$

where $u_{i,n}$ is independent of $u_{i,n+1}$; that is U_n is now a white-noise vector as required in order to apply the Kalman filter of (2.4-4). The terms $u_{i,n}$ and $u_{j,n}$ are, however, correlated for given n, as we shall see shortly. Applying the Kalman filter to a target having the target dynamics given by (2.9-7) provides the best performance in terms of minimizing the mean-square estimation error given by (2.1-2).

The transition matrix is now given by

$$\Phi(T,\tau)=\begin{bmatrix} 1 & T & \tau^2\left[-1+\dfrac{T}{\tau}+\exp\left(-\dfrac{T}{\tau}\right)\right] \\ 0 & 1 & \tau\left[1-\exp\left(-\dfrac{T}{\tau}\right)\right] \\ 0 & 0 & \exp\left(-\dfrac{T}{\tau}\right) \end{bmatrix} \tag{2.9-8}$$

When T/τ is small so the target can be considered to have a constant acceleration between sample update periods, (2.9-8) reduces to

$$\Phi(T,\tau)=\begin{bmatrix} 1 & T & \frac{1}{2}T^2 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9-9}$$

This, as expected, is identical to the transition matrix for the constant-accelerating target obtained from (1.3-1), the acceleration being constant from time n to $n+1$. The above matrix is called a Newtonian matrix [24].

The covariance of the white-noise maneuver excitation vector U_n is given by [24]

$$Q_n=E[U(n)U^T(n)]=2\alpha\sigma_a^2\begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{12} & q_{22} & q_{23} \\ q_{13} & q_{23} & q_{33} \end{bmatrix} \tag{2.9-10}$$

where

$$\alpha=\frac{1}{\tau} \tag{2.9-10a}$$

and

$$q_{11}=\frac{1}{2\alpha^5}\left(1-e^{-2\alpha T}+2\alpha T+\frac{2\alpha^3T^3}{3}-2\alpha^2T^2-4\alpha Te^{-\alpha T}\right) \tag{2.9-10b}$$

$$q_{12}=\frac{1}{2\alpha^4}\left(e^{-2\alpha T}+1-2e^{-\alpha T}+2\alpha Te^{-\alpha T}-2\alpha T+\alpha^2T^2\right) \tag{2.9-10c}$$

$$q_{13} = \frac{1}{2\alpha^3}(1 - e^{-2\alpha T} - 2\alpha T e^{-\alpha T}) \tag{2.9-10d}$$

$$q_{22} = \frac{1}{2\alpha^3}(4e^{-\alpha T} - 3 - e^{-2\alpha T} + 2\alpha T) \tag{2.9-10e}$$

$$q_{23} = \frac{1}{2\alpha^2}(e^{-2\alpha T} + 1 - 2e^{-\alpha T}) \tag{2.9-10f}$$

$$q_{33} = \frac{1}{2\alpha}(1 - e^{-2\alpha T}) \tag{2.9-10g}$$

For τ and T fixed the maneuver excitation covariance matrix Q_n is independent of n and designated as Q. Assuming again that T/τ is small, specifically $T/\tau \ll \frac{1}{2}$, done for (2.9-9), then [24]

$$Q = 2\alpha\sigma_a^2 \begin{bmatrix} \frac{1}{20}T^5 & \frac{1}{8}T^4 & \frac{1}{6}T^3 \\ \frac{1}{8}T^4 & \frac{1}{3}T^3 & \frac{1}{2}T^2 \\ \frac{1}{6}T^3 & \frac{1}{2}T^2 & T \end{bmatrix} \tag{2.9-11}$$

When T/τ is large, that is $T/\tau \gg 1$, the acceleration is independent from n, $n+1$, and

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_a^2 \end{bmatrix} \tag{2.9-12}$$

which is identical to (2.4-12), σ_a^2 being used in place σ_w^2 for the variance of the acceleration jump from time n to $n+1$.

The Kalman filter of (2.4-4) for this target model has an observation matrix given by

$$M = [1 \quad 0 \quad 0] \tag{2.9-13}$$

It is initialized using

$$x_{1,1}^* = y_1 \tag{2.9-14a}$$

$$\dot{x}_{1,1}^* = \frac{y_1 - y_0}{T} \tag{2.9-14b}$$

$$\ddot{x}_{1,1}^* = 0 \tag{2.9-14c}$$

where y_0 and y_1 are the first two range measurements. The covariance matrix

for $x^*_{n,n}$, that is, $S^*_{n,n}$, is initialized using for $n = 1$[24]

$$[S^*_{1,1}]_{00} = \sigma_X^2 \tag{2.9-15a}$$

$$[S^*_{1,1}]_{01} = [S^*_{1,1}]_{10} = \frac{\sigma_X^2}{T} \tag{2.9-15b}$$

$$[S^*_{1,1}]_{02} = [S^*_{1,1}]_{20} = 0 \tag{2.9-15c}$$

$$[S^*_{1,1}]_{11} = \frac{2\sigma_x^2}{T^2} + \frac{\sigma_a^2}{\alpha^4 T^2}\left(2 - \alpha^2 T^2 + \frac{2\alpha^3 T^3}{3} - 2e^{-\alpha T} - 2\alpha T e^{-\alpha T}\right) \tag{2.9-15d}$$

$$[S^*_{1,1}]_{12} = [S^*_{1,1}]_{21} = \frac{\sigma_a^2}{\alpha^2 T}(e^{-\alpha T} + \alpha T - 1) \tag{2.9-15e}$$

$$[S^*_{1,1}]_{22} = \sigma_a^2 \tag{2.9-15f}$$

where $[S^*_{1,1}]_{i,j}$ is the i, j element of the covariance matrix $S^*_{1,1}$ and we index the rows and columns starting with the first being 0, the second 1, and so on, to be consistent with reference 5 and Chapters 5 and 7.

If the filter acquisition occurs before the target maneuvers, that is, during a time when the target has a constant velocity, as is usually the case, then the above covariance initialization equations become simply [24]

$$[S^*_{1,1}]_{00} = \sigma_x^2 \tag{2.9-16a}$$

$$[S^*_{1,1}]_{01} = [S^*_{1,1}]_{10} = \frac{\sigma_x^2}{T} \tag{2.9-16b}$$

$$[S^*_{1,1}]_{11} = \frac{2\sigma_x^2}{T^2} \tag{2.9-16c}$$

$$[S^*_{1,1}]_{02} = [S^*_{1,1}]_{20} = [S^*_{1,1}]_{12} = [S^*_{1,1}]_{21} = [S^*_{1,1}]_{22} = 0 \tag{2.9-16d}$$

The next section gives convenient normalized design curves for the steady-state Singer *g–h–k* Kalman filters. Section 3.5.2 shows how the use of a chirp waveform affects the performance of a Singer *g–h–k* Kalman filter. Before giving the design curves we will indicate how (2.9-6) is obtained.

We start by rewriting the target dynamics equation given by (2.9-4) in state matrix form. Doing this yields

$$\dot{X}(t) = AX(t) + BN_a(t) \tag{2.9-17}$$

where

$$X(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} \tag{2.9-17a}$$

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -\alpha \end{bmatrix} \tag{2.9-17b}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.9-17c}$$

$$N_a(t) = \begin{bmatrix} 0 \\ 0 \\ n_a(t) \end{bmatrix} \tag{2.9-17d}$$

For $BN_a(t) = 0$ in (2.9-17) and $\alpha = 0$ in A of (2.9-17b) we obtain the target dynamics for the case of a constant-accelerating target with no random target dynamics. We already know the solution of (2.9-17) for this case. It is given by (2.9-6) with $U_n = 0$ and the transition matrix given by (2.9-9). To develop the solution to (2.9-17) when $BN_a(t) \neq 0$, we will make use of the solution to (2.9-17) for $BN_a(t) = 0$ obtained in Section 8.1. In Section 8.1 it is shown that the solution to (2.9-17) for $BN_a(t) = 0$ is given by

$$X(t + T) = [\exp(TA)]X(t) \tag{2.9-18}$$

see (8.1-19), where here $\zeta = T$. Alternatively, we can write

$$X(t) = [\exp(tA)]X(0) \tag{2.9-19}$$

Although A is a matrix, it is shown in Section 8.1 that $\exp(tA)$ holds with "exp = e" to a matrix power being defined by (8.1-15). We can rewrite (2.9-18) as

$$X(t_n + T) = \Phi(T)X(t_n) \tag{2.9-20}$$

or

$$X_{n+1} = \Phi X_n \tag{2.9-20a}$$

for $t = t_n$ and where

$$\Phi = \Phi(t) = \exp TA \tag{2.9-20b}$$

is the transition matrix for the target dynamics with $n_a(t) = 0$, which from (2.9-17) implies a constant-accelerating target dynamics model. For this case, as already indicated, Φ is given by (2.9-9). It is also verified in Section 8.1 that for A given by (2.9-17b) with $\alpha = 0$, (2.9-20b) becomes (2.9-9); see (8.1-10a) and problem 8.1-3. We can easily verify that for $BN_a(t) = 0$, (2.9-20) is the solution to (2.9-17) for a general matrix A by substituting (2.9-20) into (2.9-17). We carry the differentiation by treating $X(t)$ and A as if they were not matrices, in which case we get, by substituting (2.9-20) into (2.9-17),

$$d\{[\exp(tA)]X(0)\} = AX(t) \tag{2.9-21a}$$

$$[\exp(tA)]X(0)\frac{d(tA)}{dt} = AX(t) \tag{2.9-21b}$$

$$A[\exp(tA)]X(0) = AX(t) \tag{2.9-21c}$$

$$AX(t) = AX(t) \tag{2.9-21d}$$

as we desired to show.

The solution of (2.9-17) for $BN_a(t) \neq 0$ is given by the sum of the solution (2.9-20) obtained above for $BN_a(t) = 0$, which is called the homogeneous solution and which we will designate as $X(t)_H$, plus an inhomogeneous solution given by

$$W(t) = \int_0^t [\exp A(t-\varepsilon)]BN_a(\varepsilon)d\varepsilon \tag{2.9-22}$$

Thus the total solution to (2.9-17) is

$$X(t) = [\exp(At)]X(0) + \int_0^t [\exp A(t-\varepsilon)]BN_a(\varepsilon)d\varepsilon \tag{2.9-23}$$

Substituting (2.9-23) into (2.9-17) and carrying out the differentiation as done for the homogenous solution above, we verify that (2.9-23) is the solution to (2.9-17).

Comparing (2.9-23) with (2.9-6), it follows that

$$U_n = \int_{nT}^{(n+1)T} [\exp A(\tau-\varepsilon)]BN_a(\varepsilon)d\varepsilon \tag{2.9-24}$$

and $\Phi(T,\tau)$ for $\alpha \neq 0$ is given by

$$\Phi(T,\tau) = \exp AT \tag{2.9-25}$$

Substituting (2.9-17b) into (2.9-25) and using the definition for exp AT given by (8.1-15) yields (2.9-8). From (2.9-24) the independence of $U_{(n+1)}$ from U_n immediately follows and in turn (2.9-10) to (2.9-10g) follow; see [24] for details.

2.10 CONVENIENT STEADY-STATE *g–h–k* FILTER DESIGN CURVES

The constant (steady-state) *g–h–k* filter was introduced in Section 1.3. The Singer–Kalman filter described in Section 2.9 is a *g–h–k* filter. The steady-state Singer filter designs can be used as the basis for constant *g–h–k* filters. Toward this end, Fitzgerald has developed useful normalized curves that provide the steady-state weights g, h, and k for the steady-state Singer *g–h–k* Kalman filters [25]. For these curves the filter weights are given in terms of two dimensionless Fitzgerald parameters p_1 and p_2:

$$p_1 = \frac{\tau}{T} \tag{2.10-1}$$

$$p_2 = T^2 \frac{\sigma_a}{\sigma_x} \tag{2.10-2}$$

These curves are given in Figures 2.10-1 through 2.10-3. Fitzgerald also showed that the filter steady-state rms predicted and filtered position, velocity, and

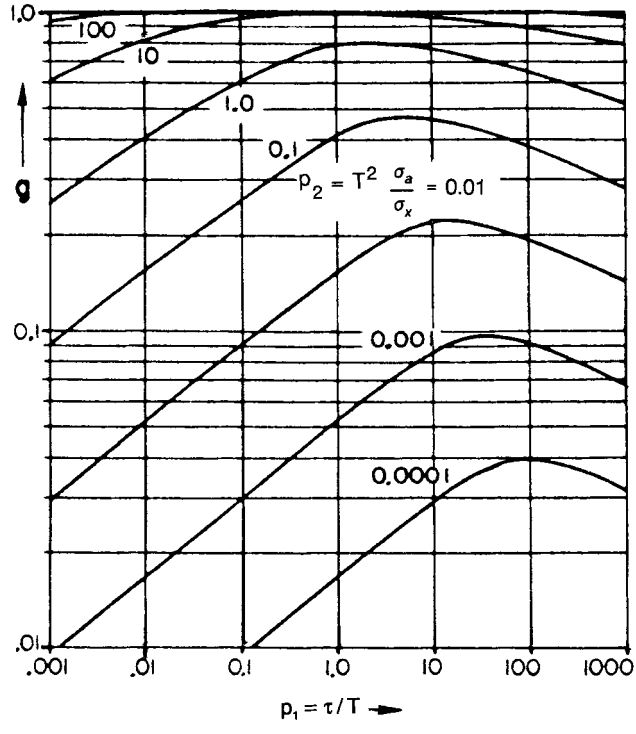

Figure 2.10-1 Fitzgerald's [25] normalized curves for g for steady-state Singer *g–h–k* filter. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

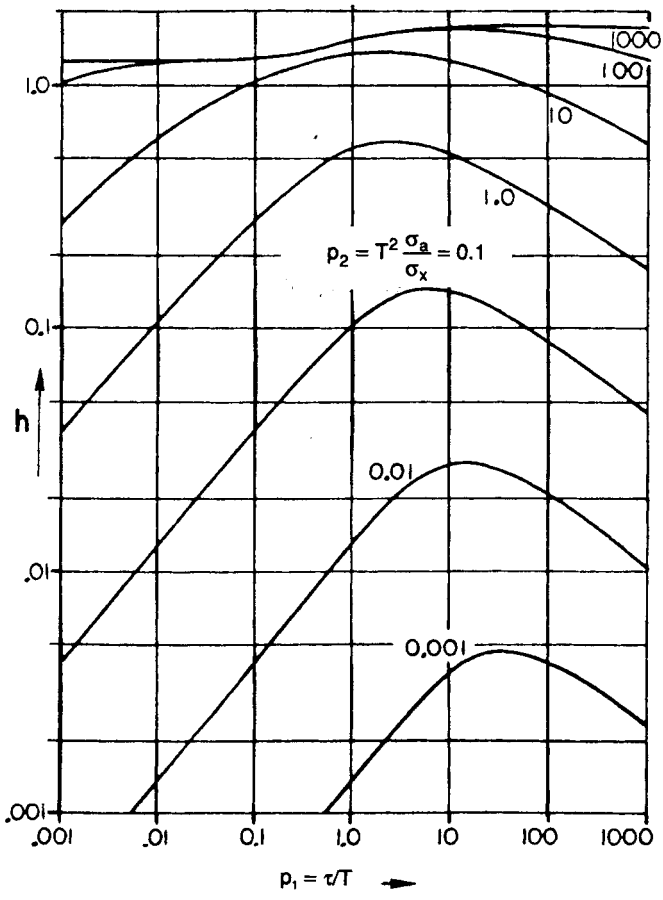

Figure 2.10-2 Fitzgerald's [25] normalized curves for h for Singer g–h–k Kalman filter. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

acceleration errors for these g–h–k filters can be expressed in terms of the two above dimensionless parameters; see Figures 2.10-4 through 2.10-7. In these figures the filtered position error is referred to by Fitzgerald as the position error after measurement update while the predicted position error is referred to as the error before the measurement update. These normalized error curves are useful for preliminary performance prediction of constant g–h–k filters.

Fitzgerald also developed normalized curves showing the accuracy the Singer filter gives for the target smoothed position near the middle of its

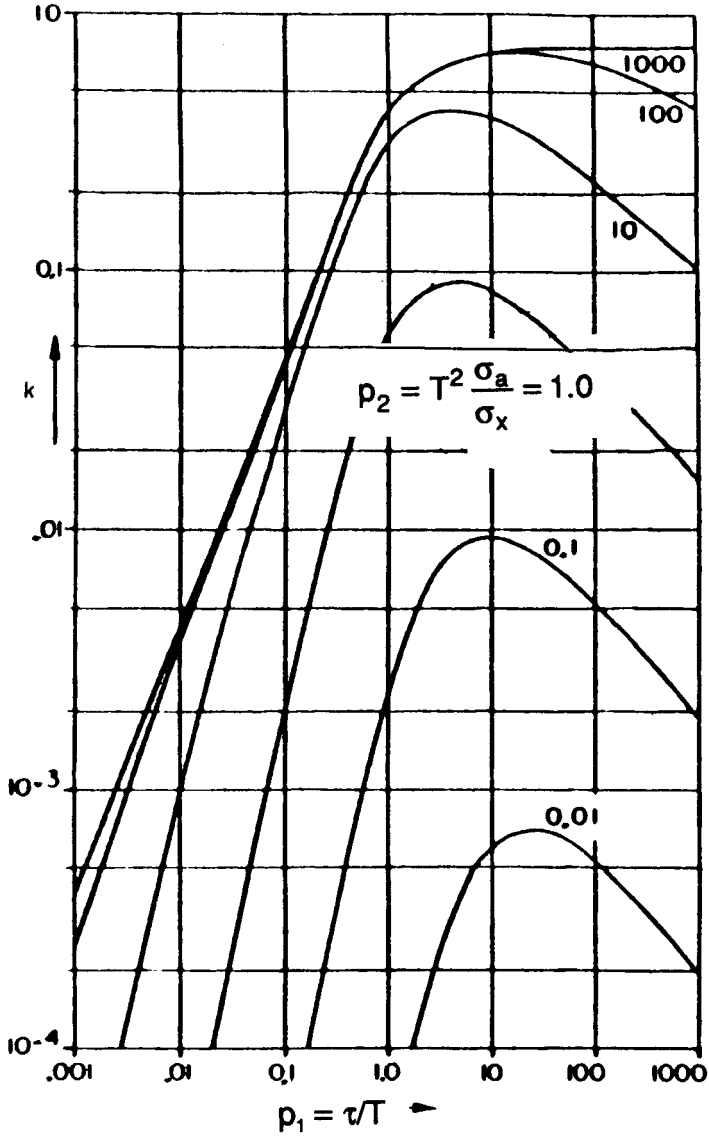

Figure 2.10-3 Fitzgerald's [25] normalized curves for k for steady-state Singer g–h–k Kalman filter. (After Fitzgerald, R. J, "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

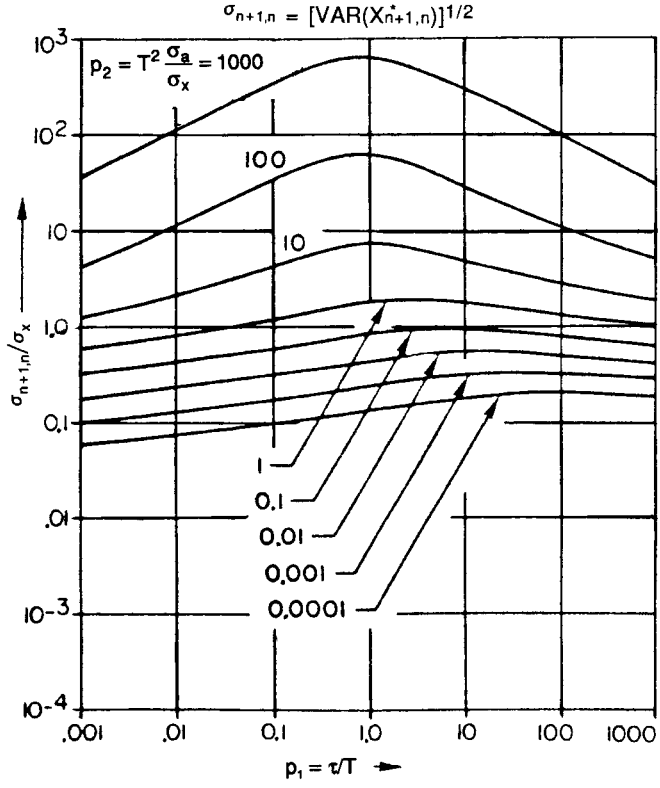

Figure 2.10-4 Fitzgerald's [25] normalized curves for one-step prediction rms error for steady-state Singer *g–h–k* Kalman filter. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

observed trajectory. Figure 2.10-8 shows these normalized curves obtained by Fitzgerald. These curves provide an indication of the improvement in the target position estimate that can be obtained post-flight after all the data have been collected. The curves apply for any point far from the endpoints of the filter-smoothing interval. Specifically, assume that the data are smoothed over the interval $n = 0, \ldots J$. Then Figure 2.10-8 applies for time n as long as $0 \ll n \ll J$. The smoothed position accuracy given in Figure 2.10-8 is normalized to the filtered target position accuracy at the trajectory endpoint $n = J$ based on the use of all the collected data.

Fitzgerald also obtained normalized curves similar to those of Figure 2.10-8 for the smoothed velocity and acceleration of the target somewhere toward the middle of the target trajectory. Figures 2.10-9 and 2.10-10 give these curves.

Physically p_1 is the random-acceleration time constant τ normalized to the track update time T. The parameter p_2 is physically 2 times the motion expected

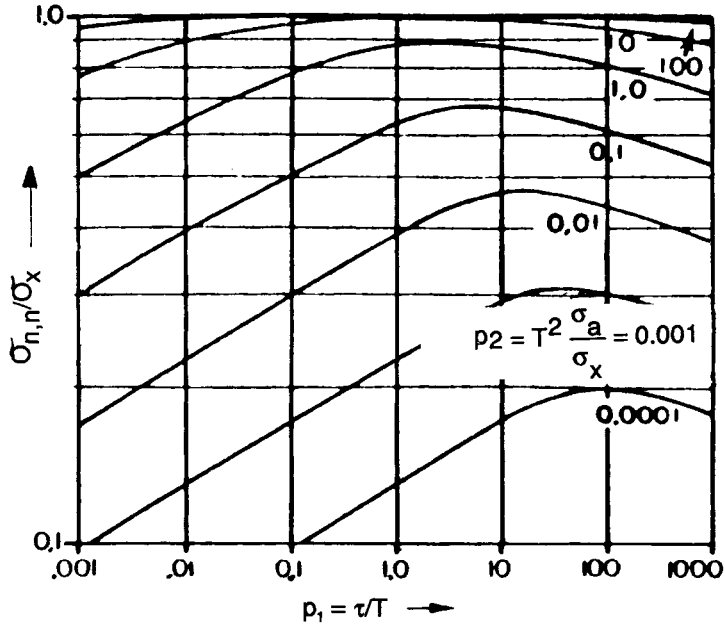

Figure 2.10-5 Fitzgerald's [25] normalized curve of rms filtered position error (i.e., position error just after update measurement has been made and incorporated into target position estimation) for steady-state Singer g–h–k Kalman filter. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

from the random acceleration $2(\frac{1}{2})\sigma_a T^2$ normalized by the measurement uncertainty σ_x. Sometimes p_2 is called the tracking index.

Example The use of the above normalized curves will now be illustrated for a g–h–k angle tracker. Assume that the target is at a slant range $R = 50$ km; that the target is characterized by having $\tau = 3$ sec and $\sigma_a = 30\,\text{m/sec}^2$, and that the radar 3-dB beamwidth $\theta_3 = 20$ mrad and the angle measurement error $\sigma_\theta = 1$ mrad, θ replacing x here in Figures 2.10-4 to 2.10-10. Assume that the 3σ predicted tracking position in angle is to be less than $\frac{1}{2}\theta_3$ for good track maintenance. The problem is to determine the tracker sampling rate required, the tracker rms error in predicting position, the ability of the tracker to predict position after track update (the target's filtered position), and the ability of the tracker to predict the target rms position by postflight smoothing of the data (to some point far from the trajectory endpoints, as discussed above).

Solution: We desire the predicted angle normalized position error $\sigma_{n+1,n}(\theta)$, which we here disignate as $\sigma_{n+1,n}$ for convenience [noting that this term serves

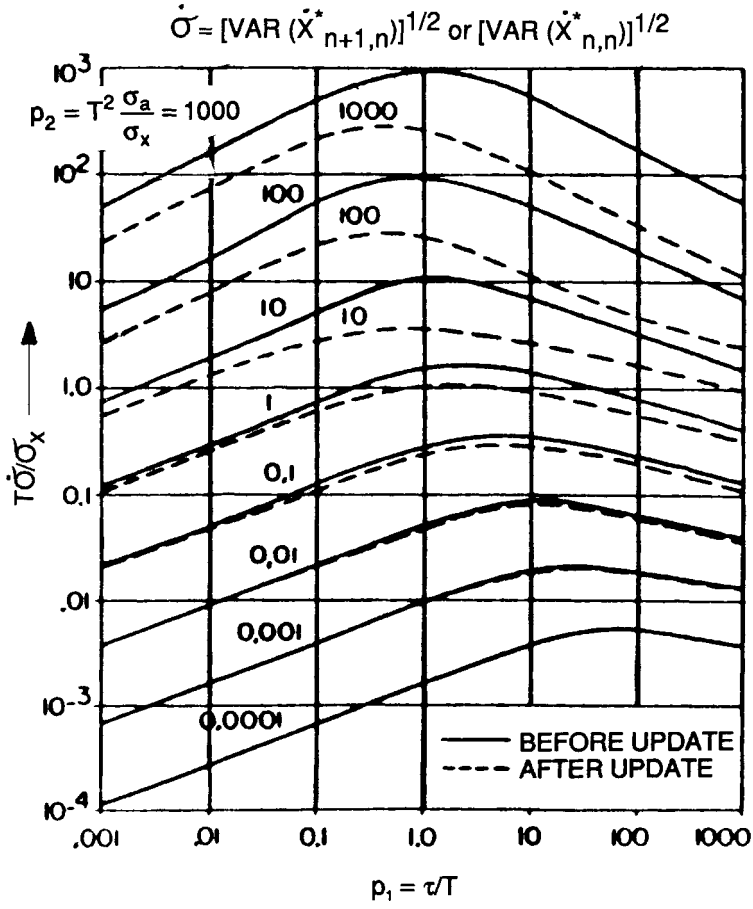

Figure 2.10-6 Fitzgerald's [25] normalized curves for one-step prediction and filtered velocity rms errors (i.e., before and after update respectively) for steady-state Singer *g–h–k* Kalman filter. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

double duty; see (1.2-23)], to be given by

$$3\sigma_{n+1,n} \leq (\tfrac{1}{2})\,\theta_3 = 20\,\text{mrad}/2 = 10\,\text{mrad} \tag{2.10-3}$$

Normalizing the above by dividing both sides by $3\sigma_\theta$ yields

$$\frac{\sigma_{n+1,n}}{\sigma_\theta} \leq \frac{\theta_3}{2}\frac{1}{3(\sigma_\theta)} = \frac{10\,\text{mrad}}{3(1\,\text{mrad})} = 3.33 \tag{2.10-4}$$

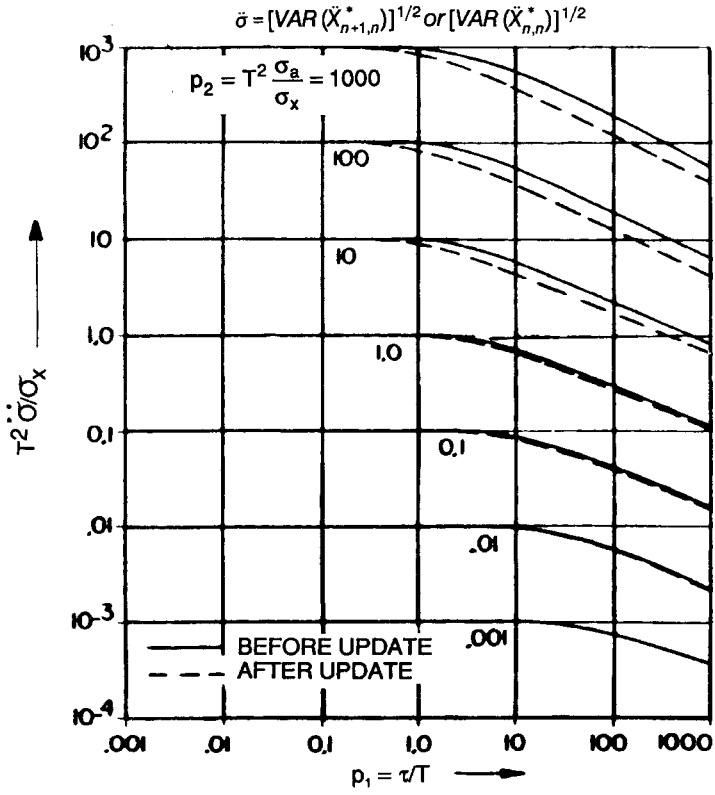

Figure 2.10-7 Fitzgerald's [25] normalized curves for one-step prediction and filtered acceleration rms errors (i.e., before and after update respectively) for steady-state Singer *g–h–k* Kalman filter. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

In Figure 2.10-4, p_1 and p_2 are not known. Also, because T is in both p_1 and p_2, an iteration trial-and-error process is needed generally to solve for T. A good starting point is to assume that the solution is at the peak of a $p_2 =$ constant curve. As a result it follows from Figure 2.10-4 that if (2.10-4) is to be true it is necessary that $p_2 \leq 2.5$. As a result

$$p_2 = T^2 \frac{\sigma_a}{\sigma_{cx}} = T^2 \frac{30\,\text{m/sec}^2}{(1\,\text{mrad})\,(50\,\text{km})} \leq 2.5 \tag{2.10-5}$$

where σ_{cx} is the cross-range position rms measurement accuracy given by

$$\sigma_{cx} = \sigma_\theta R = (1\,\text{mrad})\,(50\,\text{km}) = 50\,\text{m} \tag{2.10-6}$$

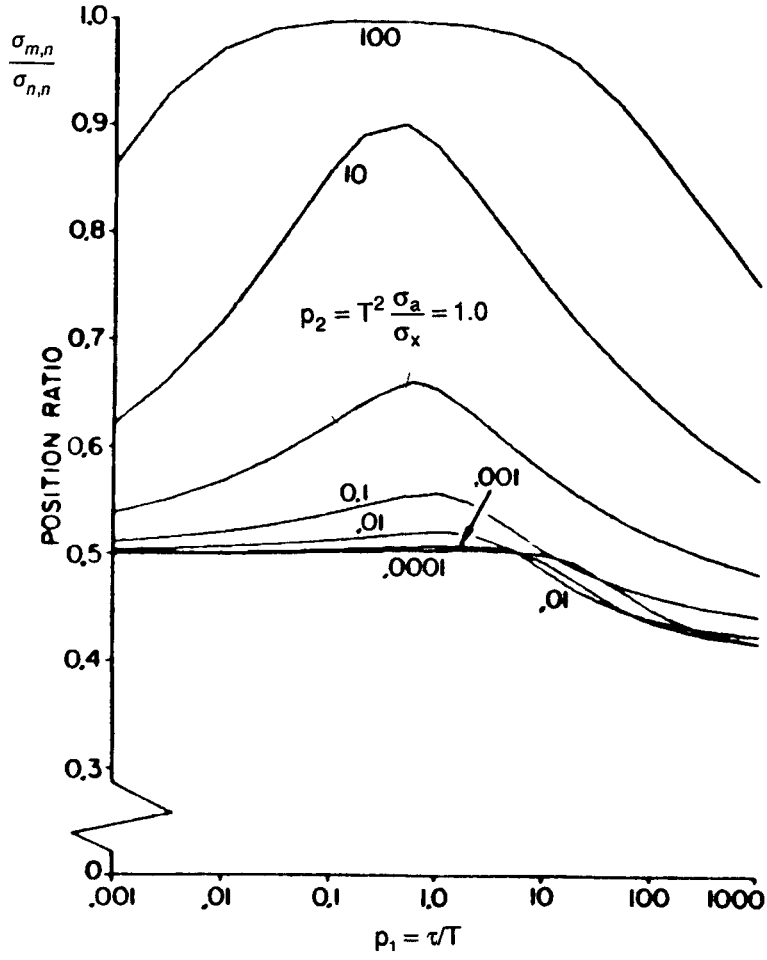

Figure 2.10-8 Fitzgerald's [25] curves for normalized accuracy to which target's position can be obtained for observed point somewhere in middle of target track history, the smoothed position accuracy. Normalization is with respect to steady-state filtered position error $\sigma_{n,n}$ given in Figure 2.10-5. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transactions on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

From the above (2.10-5) it follows that $T \leq 2.04$ sec. We will choose $T = 2$ sec as our first iteration estimate for T.

For the above value of T it follows that $p_1 = \tau/T = (3\,\text{sec})/(2\,\text{sec}) = 1.5$. We now check if this value for p_1 is consistent with the value for p_1 obtained from Figure 2.10-4 for $p_2 = 2.5$. If it is not, then another iteration is required in

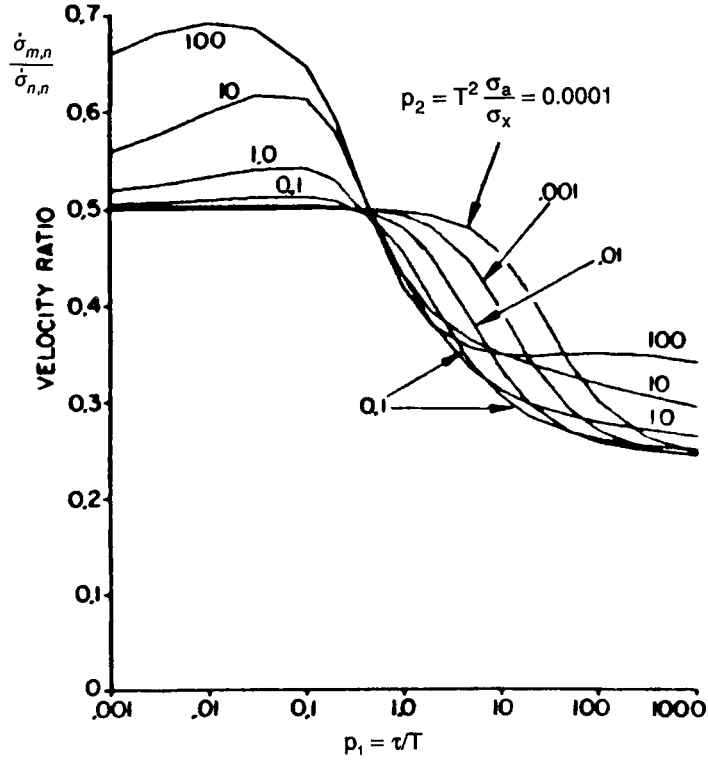

Figure 2.10-9 Fitzgerald's [25] normalized curves for accuracy to which target's velocity can be obtained at point somewhere in middle of target track history. Normalization is with respect to filtered velocity estimate given in Figure 2.10-6. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transaction on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

the selection of T. Figure 2.10-4 yields a value of $p_1 = 1.5$. Hence the value for p_1 obtained from Figure 2.10-4 is consistent with the value obtained for $T = 2$ sec. As a result a second iteration is not needed for our example. In general, the requirement for an iteration being carried out in order to solve for T could be eliminated if the curve of Figure 2.10-4 is replaced with a different normalized curve, specifically, if p_2 is replaced using curves of constant $\tau^2\sigma_a/\sigma_x$ [25].

Figure 2.10-5 yields the normalized after-measurement (filtered) estimate of the target position as 0.93. Hence the unnormalized estimate of the filtered target position is given by 0.93 (50 m) = 46.5 m. From Figure 2.10-8 it follows that the postflight rms estimate of position is given by 0.73(46.5) = 34 m.

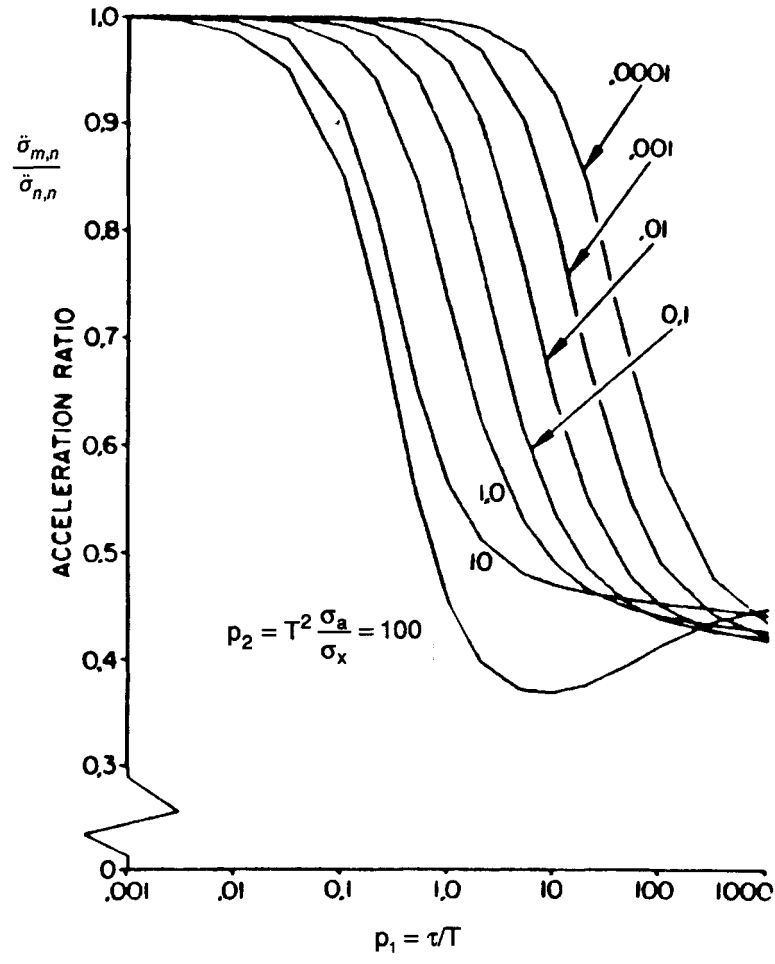

Figure 2.10-10 Fitzgerald's normalized curves providing accuracy to which target's acceleration can be obtained for point somewhere in middle of target track history. Normalization is with respect to filtered acceleration error given in Figure 2.10-7. (After Fitzgerald, R. J., "Simple Tracking Filters: Steady-State Filtering and Smoothing Performance," IEEE Transaction on Aerospace and Electronic Systems, Vol. AES-16, No. 6, November 1980. © 1980 IEEE.)

2.11 SELECTION OF TRACKING FILTER

Generally one would like to use the Kalman filter because it gives the best performance. However, the Kalman filter also imposes the greatest computer complexity. Table 2.11-1 gives a comparison of the accuracy of five tracking filters as obtained for various missions in reference 26. The Kalman filter is seen to provide the best performance. The two-point extrapolator uses the last

TABLE 2.11-1. Synopsis of the Accuracy Comparison of the Five Tracking Filters (After Singer and Behnke [26])

	Target Type	Air			Surface or Subsurface
Filter type	Sensor Type	Air Search Radar	Surface and Air Search Radar	Surface and Air Search Radar	Radar, Sonar
Two-point extrapolator		3	3	3	3
Wiener filter		0	0	0	0
g–h Filter		2	0	2	1
Simplified Kalman filter		0	0	0	0
Kalman filter		0	0	0	0

Note: 0 = within 20% of Kalman filter; 1 = 20–40% worse than Kalman filter; 2 = 40–70% worse than Kalman filter; 3 = more than 70% worse than Kalman filter.

TABLE 2.11-2. Comparison of the Computer Requirements for the Five Filters

	Initialization		Main Loop	
Filter Type	Time[a]	Memory Locations[b]	Time[a]	Memory Location[b]
Two-point extrapolator	7	15	7	15
Wiener filter	8	29	21	33
g–h Filter	40	46	44	58
Simplified Kalman filter	51	54	81	71
Kalman Filter	54	67	100	100

[a] Percentage of the computer time required by the Kalman filter in the main loop.
[b] Percentage of the memory locations required by the Kalman filter in the main loop.
Source: From Singer and Behnke [26].

received data point to determine the target range and bearing and the last two data points to determine the target range-rate and bearing rate. This is the simplest filter for estimating target kinematics, Table 2.11-2 gives a comparison of the computer time and memory requirements for the five filters. All the filters of Table 2.11-1 except the two-point extrapolator met the system requirements (of reference 26) relative to prediction accuracy. The constant *g–h* filter meets the accuracy requirements and at the same time requires very little computer time. Hence this filter might be selected. However, if the radar is in addition required to calculate weapon kill probabilities, then the constant *g–h* filter would probably not be used. This is because this filter does not by itself provide inputs for the calculation of weapon kill probabilities (like accuracies of the

TABLE 2.11-3. Filter and System Parameters for the Second Example

	Prediction Accuracy (1σ)				Percent of Computer Time Devoted to Tracking	Auxiliary Functions
	Range (yd)	Bearing (deg)	Speed (knot)	Course (deg)		
System requirements	150	0.80	2.5	7.0	10	Kill probability calculations
Two-point extrapolator	210	1.04	3.7	9.7	0.8	—
Wiener Filter	142	0.71	2.4	6.8	2.4	—
α–β Filter	173	0.80	2.8	8.2	5.2	—
Simplified Kalman filter	138	0.65	2.4	6.5	9.5	[a]
Kalman filter	130	0.63	2.1	6.1	12	[a]

[a] Tracking accuracy statistics suitable for the auxiliary functions are calculated automatically.
Source: From Singer and Behnke [26].

target predicted location). On the other hand, the Kalman filter provides the accuracy of the predicted position of the target as a matter of course in computing the filter weights. For the constant *g–h* filter an additional computation has to be added to the standard filter computations. With these additional computations using the constant *g–h* filter requires as many computations as does the simplified Kalman filter. As a result, the simplified Kalman filter would be preferred for the application where weapon kill probabilities have to be calculated, this filter providing tracking accuracies within 20% of the Kalman filter. The Kalman filter was not selected because its computer time computations were greater than specified by the system requirements for the application of reference 26 and for the time at which the work for reference 26 was performed (prior to 1971). With the great strides made in computer throughput and memory over the last two decades the choice of filter today may be different. Table 2.11-3 gives the comparison of the performance of the various filters for a second example of reference 26.

Another problem with using the constant *g–h* filter is that it does not provide good performance when the target is maneuvering. However, aircraft targets generally go in straight lines, rarely doing a maneuver. Hence, what one would like to do is to use a Kalman filter when the target maneuvers, which is rarely, and to use a simple constant *g–h* filter when the target is not maneuvering. This can be done if a means is provided for detecting when a target is maneuvering. In the literature this has been done by noting the tracking-filter residual error, that is, the difference between the target predicted position and the measured position on the *n*th observation. The detection of the presence of a maneuver

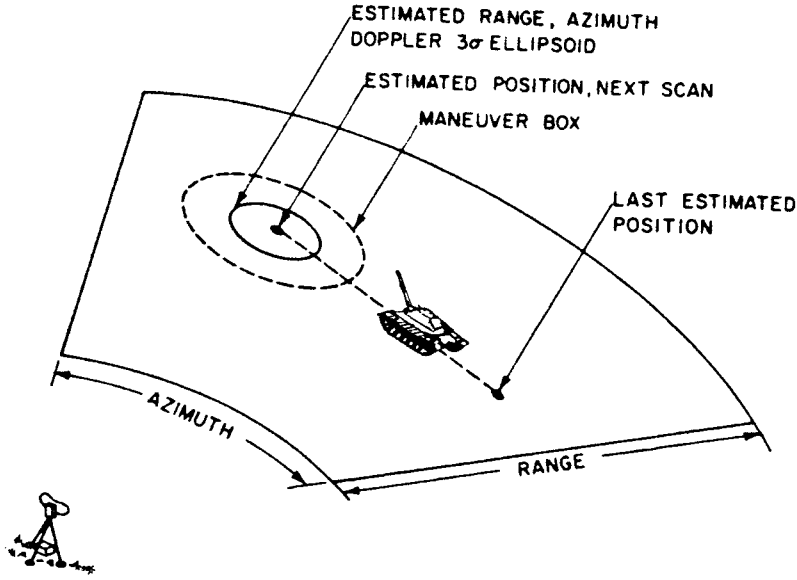

Figure 2.11-1 Nonmaneuvering and maneuvering detection windows. (From Mirkin et al., "Automated Tracking with Wetted Ground Surveillance Radars," Proceedings of 1980 IEEE International Radar Conference. © 1980 IEEE.)

could be based either on the last residual error or some function of the last m residual errors.

An alternate approach is to switch when a maneuver is detected from a steady-state g–h filter with modest or low g and h values to a g–h filter with high g and h values, similar to those used for track initiation. This type of approach was employed by Lincoln Laboratory for its netted ground surveillance radar system [27]. They used two prediction windows to detect a target maneuver, see Figure 2.11-1. If the target was detected in the smaller window, then it was assumed that the target had not maneuvered and the values of g and h used were kept at those of the steady-state g–h filter. Specifically g and h were selected to be 0.5 and 0.2, respectively, in reference 27. If the target fell outside of this smaller 3σ window but inside the larger window called the maneuver window, the target was assumed to have maneuvered. In this case the weights g and h of the filter were set equal to 1.0, thus yielding a filter with a shorter memory and providing a better response to a maneuvering target.

Even when using a Kalman filter, there is still the question as to what value to use for the variance of the maneuver σ_a^2 of (2.9-1). Lincoln Laboratory evaluated a Kalman filter that used $\sigma_a = 0$ when the target was not maneuvering and a nonzero value when a maneuver was detected [28]. A weighted sum of the residuals from the previous observations was used for detecting the presence of a maneuver. When a maneuver was detected, the appropriate change on the filter driving noise variance was made and the filter

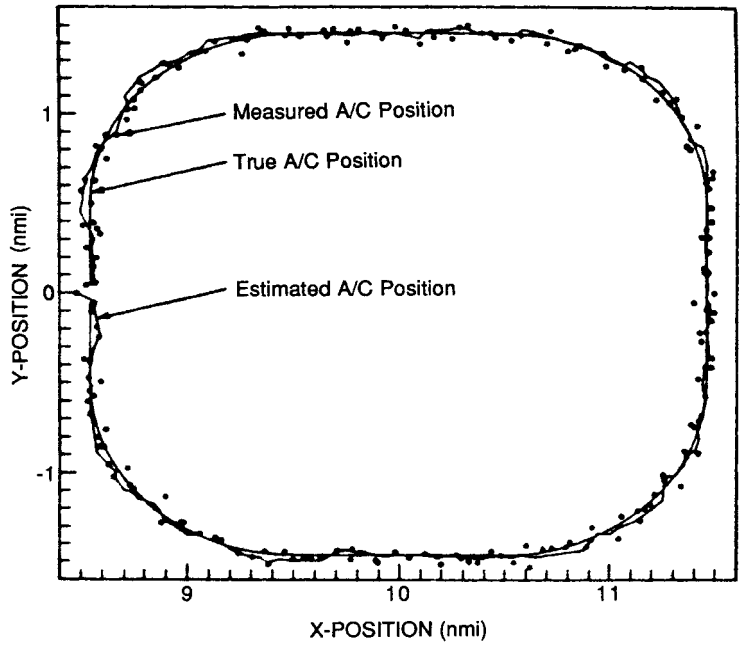

Figure 2.11-2 Performance of Kalman filter in tracking aircraft in holding pattern. Target periodically goes into turn maneuver followed by straight-line trajectory. (From McAulay and Denlinger, "A Decision Directed Adaptive Tracker," IEEE Transactions on Aerospace and Electronic Systems, March 1973. © IEEE.)

reinitialized based on the most recent measurements. Figure 2.11-2 shows how well the filter worked when tracking an aircraft in a holding pattern and doing turns as indicated. The radar was assumed to be 10 miles from the center of the holding pattern shown in Figure 2.11-2 and assumed to have an accuracy of 150 ft in range and 0.2° in angle. The target was assumed to be flying at 180 knots when flying in a straight line. This filter was intended for an air traffic control (ATC) application. Its performance was evaluated by seeing how well it could maintain track on an aircraft in a modified holding pattern involving 90° turns at 3° per second, as shown in Figure 2.11-2. It is seen that the filter worked very well.

Another approach to handling changing target dynamics is to continually change the Kalman filter dynamics based on the weighted average of the most recent residual errors. This approach was used by the Applied Physics Laboratory (APL) [17]. Still other approaches are discussed elsewhere [29].

A final comparison obtained from reference 8 between the performance of a g–h Kalman filter and constant g–h filters will now be given. The g–h Kalman filter design was obtained for the dynamics model given by (2.1-1) or (2.4-2) and (2.4-2a) for which the maneuver covariance excitation matrix is given by (2.4-10). The Kalman filter design was obtained assuming $\sigma_u = 1\,\text{m/sec}$ for

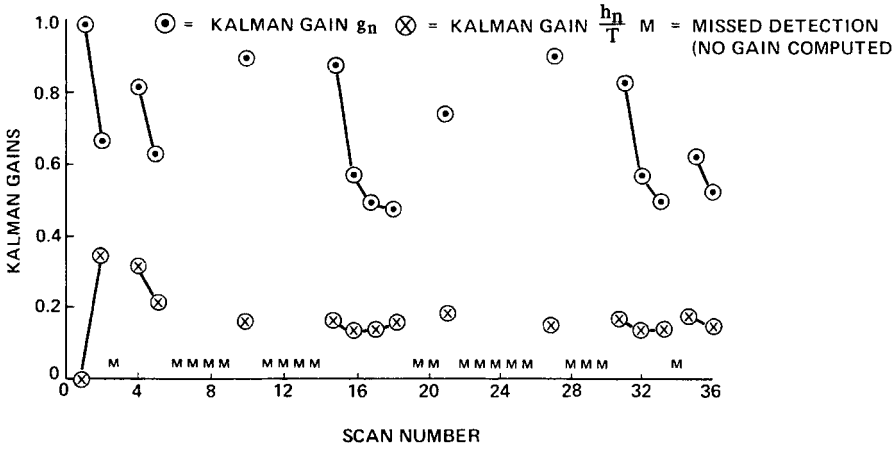

Figure 2.11-3 Two-state Kalman filter simulation; $T = 1$ sec, hence Kalman gain $h_n/T = h_n$. (Reprinted with permission from *Multiple-Target Tracking with Radar Applications*, by S. S. Blackman. Artech House, Inc., Norwood, MA, USA, 1986.)

(2.4-10), $\sigma_x = 5$ m and $T = 1$ sec. The filter was initiated assuming the rms of $x^*_{0,0}$ and $\dot{x}^*_{0,0}$ are 10 m and 5 m/sec, respectively. It was also assumed that a track detection was achieved on each revisit with probability 0.5. Figure 2.11-3 shows the results of a simulation for this Kalman filter. The position gain g_n starts high (0.997) and then oscillates between about 0.5 and 0.9. The velocity gain h_n starts at zero, goes to 0.3 for the next two observations, and then remains at about 0.15. The effects of missed detections on g_n and h_n is clearly indicated in Figure 2.11-3, g_n in increasing significantly after missed detections.

Based on the results of Figure 2.11-3 a constant *g–h* filter was designed as an approximate replacement for the Kalman filter [8]. For this approximate filter $g = 0.7$ and $h = 0.15$ were chosen. It was found that the approximate constant-gain filter had a $\sigma_{n+1,n}$ that was only about 5% larger than obtained for the Kalman filter. For a larger σ_u, $\sigma_{n+1,n}$ increased at least 10%. It was found that using a Benedict–Bordner filter design or a *g–h* filter design based on reference 15 gave severely degraded performance. The difference is that the Kalman filter design, and as a result its approximate replacement, picked g and h to compensate for missed data. Hence if a constant *g–h* filter or *g–h–k* filter is to be used for a tracking application where there can be missed updates with high probability, the weights should be determined taking into account the possibility of missed updates as done above.

With the enormous advances made in computer throughput, some have argued that *g–h* and *g–h–k* filters should never be used, that we should now use just Kalman filters [30,31]. This point of view may be a bit extreme. Reference 8 argues for using simplifications and approximations to the Kalman filter in an

attempt to reduce computational requirements significantly without degrading tracking performance. One technique is the use of a table lookup for the weights, as described in Section 2.7. Another is the approximation of the Kalman filter weights using fixed weights that account for the possibility of missed updates, as done above for the example of Figure 2.11-3. With this approach the full-blown Kalman filter is simulated with missed updates to determine the approximate steady-state weights to use. Still another approach is the use of a lower order filter, such as a *g–h* instead of a *g–h–k* [8]. The reader is referred to reference 8 for further discussions on obtaining approximations to the Kalman filter. Certainly studying and designing *g–h* and *g–h–k* filters for a given application will give a physical feel for what to expect from a Kalman filter.