9

PRACTICAL CONSIDERATIONS

> The classical (more accurately, old-fashioned) view is that a mathematical problem is solved if the solution is expressed by a formula. It is not a trivial matter, however, to substitute numbers in a formula[1].

9.1 CHAPTER FOCUS

The discussion turns now to what might be called *Kalman filter engineering*, which is that body of applicable knowledge that has evolved through practical experience in the use and misuse of the Kalman filter. The material of the previous two chapters (square-root and nonlinear filtering) has also evolved in this way and is part of the same general subject. Here, however, the discussion includes many more matters of practice than nonlinearities and finite-precision arithmetic.

9.1.1 Main Points to Be Covered

1. *Roundoff Errors are Not the Only Causes for the Failure of the Kalman Filter to Achieve Its Theoretical Performance*. There are diagnostic methods for identifying causes and remedies for other common patterns of misbehavior.

[1]R. E. Kalman and R. S. Bucy, "New results in linear filtering and prediction theory," Journal of Basic Engineering, Series D, Vol. 83, pp. 95–108, 1961.

Kalman Filtering: Theory and Practice Using MATLAB®, Fourth Edition.
Mohinder S. Grewal and Angus P. Andrews.

2. *Prefiltering to Reduce Computational Requirements*. If the dynamics of the measured variables are "slow" relative to the sampling rate, then a simple prefilter can reduce the overall computational requirements without sacrificing performance.
3. *Detection and Rejection of Anomalous Sensor Data*. The inverse of the matrix $(HPH^{\mathrm{T}} + R)$ characterizes the probability distribution of the innovation $z - H\hat{x}$ and may be used to test for exogenous measurement errors, such as those resulting from sensor or transmission malfunctions.
4. *Statistical Design of Sensor and Estimation Systems*. The covariance equations of the Kalman filter provide an analytical basis for the predictive design of systems to estimate the state of dynamic systems. They may also be used to obtain suboptimal (but feasible) observation scheduling.
5. *Testing for Asymptotic Stability*. The relative robustness of the Kalman filter against minor modeling errors is due, in part, to the asymptotic stability of the Riccati equations defining performance.
6. *Model Simplifications to Reduce Computational Requirements*. A dual-state filter implementation can be used to analyze the expected performance of simplified Kalman filters, based on simplifying the dynamic system model and/or measurement model. These analyses characterize the trade-offs between performance and computational requirements.
7. *Memory and Throughput Requirements*. These computational requirements are represented as functions of "problem parameters" such as the dimensions of state and measurement vectors.
8. *Offline Processing to Reduce Online Computational Requirements*. Except in extended (nonlinear) Kalman filtering, the gain computations do not depend upon the real-time data. Therefore, they can be precomputed to reduce the real-time computational load.
9. *Innovations Analysis*. It is a rather simple check for symptoms of mismodeling.

9.2 DIAGNOSTIC STATISTICS AND HEURISTICS

This is a collection of methods that have been found useful in understanding the observed behavior of Kalman filters and for detecting and correcting anomalous behavior.

9.2.1 Innovations Analysis

Innovations[2] are the differences between observed and predicted measurements,

$$\nu_k \stackrel{\text{def}}{=} z_k - H_k \hat{x}_{k(-)}. \tag{9.1}$$

[2]Kailath [1] introduced the notation ν (Greek letter "nu") for innovations, because they represent "what is new" in the measurement.

Innovations are the carotid artery of a Kalman filter. They provide an easily accessible point for monitoring vital health status without disrupting normal operations, and the statistical and temporal properties of its pulses can tell us much about what might be right or wrong with a Kalman filter implementation.

9.2.1.1 Properties of Innovations If the Kalman filter is properly modeled for its task, its innovations

1. Have zero mean

$$\underset{k}{\mathrm{E}}\langle \nu_k \rangle = 0; \tag{9.2}$$

2. Are white (i.e., uncorrelated in time)

$$\underset{k \neq j}{\mathrm{E}} \langle \nu_k \nu_j^{\mathrm{T}} \rangle = 0; \tag{9.3}$$

3. Have known covariance

$$P_{\nu\nu k} \overset{\text{def}}{=} \mathrm{E}_k \langle \nu_k \nu_k^{\mathrm{T}} \rangle \tag{9.4}$$

$$= H_k P_{k(-)} H_k^{\mathrm{T}} + R_k; \tag{9.5}$$

4. Have known information matrix

$$Y_{\nu\nu k} \overset{\text{def}}{=} P_{\nu\nu k}^{-1} \tag{9.6}$$

$$= [H_k P_{k(-)} H_k^{\mathrm{T}} + R_k]^{-1}, \tag{9.7}$$

   which is a partial result in the computation of the Kalman gain;

5. Have a known mean value for the information quadratic form

$$\underset{k}{\mathrm{E}} \langle \nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}} \rangle = \ell, \tag{9.8}$$

   the dimension of the measurement vector z_k;

6. And—if all the error sources are Gaussian—the information quadratic forms have a chi-squared distribution with ℓ degrees of freedom

$$\{ \nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}} \} \in \chi_\ell^2. \tag{9.9}$$

Likely causes for anomalous values for these innovations statistics are addressed in the following subsections.

Isolating the cause often comes down to deciding whether the source is the sensor (z_k), the dynamic system model ($H_k \hat{x}_{k(-)}$), or exogenous sources. In either case, it is cause for reexamination of the entire Kalman filtering model and for questioning whether the cause could be some exogenous source such as power supply noise,

mechanical vibrations induced by environmental conditions, or diurnal variations in light levels, temperature, and humidity. Even cyclic variations due to heating, ventilation, and air conditioning systems have been known to create noticeable errors. Analysis of innovations may give some clues about likely source(s).

In all cases, any changes made to correct anomalous behavior detected from innovations diagnostics must be verified by follow-up evaluations of the same diagnostics to confirm the diagnosis.

Example 9.1 (Simple Innovations Test of Mismodeling) As a simple illustration of the general approach, the MATLAB® file `InnovAnalysis.m` in the Wiley web site simulates a linear stochastic process with nine different Kalman filters: one with the same model parameters as the simulated process and eight with the four model parameters Φ, H, Q, and R scaled up and down by a factor of two. The empirical mean values of the χ^2 statistics are then compared. In addition, because the process is being simulated, one can calculate the simulated filter performances in terms of the root-mean-square (RMS) differences between the simulated state variable and the estimates from the nine Kalman filters.

The results of a Gaussian Monte Carlo simulation with a 1000 time steps are summarized in Table 9.1. These do indicate that the correctly modeled Kalman filter has a χ^2 statistic close to $\ell = 1$ and significant deviations of the same χ^2 statistics for the eight different modeling deviations. Some mean χ^2 values are larger than those from the correctly modeled case, and some are smaller. In all cases, the a priori and a posteriori RMS estimation accuracies of the mismodeled Kalman filters implementations are worse than those of the correctly modeled Kalman filter—but one cannot know the true state variables except in simulation. The χ^2 statistics, on the other hand, can always be calculated from the Kalman filter parameter values and the filter innovations.

These results do indicate that—for this particular application implementation—the innovations mean is a reasonably good indicator of mismodeling of some sort. Changing a parameter by a factor of two may not change the chi-squared mean by a factor of two, but it does change it noticeably.

TABLE 9.1 Example Innovations Analysis of Mismodeled Implementations

Kalman Filter Parameter Values*				Performance		
				A Priori	A Posteriori	χ^2 Mean
Φ	H	Q	R	0.4813	0.2624	0.9894
2Φ	H	Q	R	1.4794	0.5839	3.5701
$\Phi/2$	H	Q	R	0.7237	0.4893	1.9269
Φ	$2H$	Q	R	0.6762	0.5938	0.3843
Φ	$H/2$	Q	R	0.9637	0.9604	1.9526
Φ	H	$2Q$	R	0.6133	0.5015	0.6411
Φ	H	$Q/2$	R	0.6036	0.4860	1.4248
Φ	H	Q	$2R$	0.6036	0.4861	0.7130
Φ	H	Q	$R/2$	0.6133	0.5015	1.2810

*Simulated process model parameters are Φ, H, Q, and R.

9.2.1.2 Diagnosing Means Not-uncommon causes for the means of innovations sequences being nonzero include the following:

1. Nonzero-mean sensor noise, also called *sensor bias error*. If the bias is constant, this can be verified and corrected by recalibration of the sensor(s) involved. Otherwise, sensor bias can be appended to the state vector as an exponentially correlated process or a random walk.
2. Overrated sensor noise, in which the value of R has been set too high. When the state variables are unknown constants, this can cause the Kalman gain to be too small, which causes delayed convergence. This, too can be verified and corrected by recalibration of the sensor(s) or by appending elements of R to the state vector as unknown parameters.
3. Underrated dynamic disturbance noise, which can also lead to lagged convergence. "Tuning" the dynamic disturbance noise covariance is a fairly common remedy for apparent slow convergence, and this is sometimes implemented as a parameter estimation problem. In some cases, however, this may only be masking other mismodeling errors.

9.2.1.3 Autocorrelation Analysis The MATLAB function `xcov` in the Signal Processing Toolbox can be used for computing the autocovariance of innovations, which should be near zero except at zero lag. The zero-lag value should equal P_{vv}, as given by Equation 9.5.

If the zero-lag value is quite different from P_{vv}, there are a number of options, including "tuning" Q or R.

9.2.1.4 Spectrum Analysis This is especially useful for detecting unmodeled resonances, including unmodeled harmonics in sensor noise and any harmonics of the operational environment. The power spectral densities and cross-spectral densities can be computed by taking the fast Fourier transform of the autocovariances from the previous subsection or by computing the spectra and cospectra directly from innovations sequences.

The frequencies of detected harmonics can be useful for deciding whether to look for electronic, vibrational, thermal control, or diurnal variations as likely sources.

If the corrupting harmonics occur only on a single sensor output, that sensor may be suspect. In that case, the sensor noise model can be augmented to include the harmonic noise.

If the same harmonic frequency is spread across multiple outputs, then the inverse sensor transformation $H_k^\dagger$ (Moore–Penrose inverse of H_k) may offer a clue of the system-level source. If, for example, $H_k^\dagger$ maps the problem back to a single state variable, the dynamic model of that state variable can be augmented to include the offending harmonic term.

The center of a harmonic peak identifies the frequency of the missing harmonic model, and the spreading of the harmonic peak may give some indication of the modeled damping factor to be used.

9.2.1.5 Covariance/Information Analysis The analysis described in Section 9.2.1.3 results in a statistic that should resemble $P_{\nu\nu}$, which is computed as a partial result in the calculation of Kalman gain.

If not, then either R_k or $H_k P_{k(-)} H_k^{\mathrm{T}}$ may be blamed and either can be tuned accordingly.

9.2.1.6 Chi-Squared Means The mean value of the innovations-norm sequence

$$\{\nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}}\}$$

should equal the dimension ℓ of the measurement vector z_k.

If it is larger, then either R_k or $H_k P_{k(-)} H_k^{\mathrm{T}}$ may be blamed, and either can be tuned accordingly.

9.2.1.7 Chi-Squared Distributions The covariance of a chi-squared distribution should be twice its mean

$$\mathrm{E}_k \langle [\nu_k Y_{\nu\nu k} \nu_k^{\mathrm{T}} \ell]^2 \rangle = 2\ell, \tag{9.10}$$

which provides another indicator of whether the distribution of innovations norms is actually chi-squared.

However, the Kalman filter seems to operate well even when the underlying random processes are not necessarily Gaussian, so the discovery that the histograms of an innovations sequence do not look like a chi-squared distribution should perhaps be taken with a grain of salt.

Example 9.2 (Sensor Roundoff Errors) Roundoff errors are ubiquitous in Kalman filter implementations. As a rule, they cannot be monitored because they are not detectable within the precision limits of the implementation—except when the least significant bit (LSB) of a digitized sensor output is bigger than the LSB of the processor (which is not uncommon). In that case, sensor noise can be dominated by roundoff error, which is decidedly non-Gaussian.

In a well-designed digitizer, roundoff errors $\varepsilon_{\mathrm{rnd}}$ tend to be uniformly distributed from $-1/2$ LSB to $+1/2$ LSB. That is $\varepsilon_{\mathrm{rnd}} \in \mathcal{U}([-1/2 \text{ LSB}, +1/2 \text{ LSB}])$, the uniform distribution from $-1/2$ LSB to $+1/2$ LSB. The mean of this distribution is zero and its variance and information will be

$$\begin{aligned}
\sigma_{\mathrm{rnd}}^2 &\stackrel{\mathrm{def}}{=} \mathop{\mathrm{E}}_{\varepsilon_{\mathrm{rnd}}} \langle \varepsilon_{\mathrm{rnd}}^2 \rangle \\
&= \frac{1}{\mathrm{LSB}} \int_{-1/2 \text{ LSB}}^{+1/2 \text{ LSB}} \varepsilon^2 \, d\varepsilon \\
&= \frac{1}{\mathrm{LSB}} \left[\frac{\varepsilon^3}{3} \right]_{-1/2 \text{ LSB}}^{+1/2 \text{ LSB}} \\
&= \mathrm{LSB}^2/12
\end{aligned}$$

$$Y_{\text{rnd}} = 12/\text{LSB}^2,$$

respectively. Sensor roundoff errors could then dominate the innovations

$$\begin{aligned} &\text{if } R \approx \text{LSB}^2/12 \\ &\text{and } R \gg HPH^{\text{T}} \\ &\text{then } \nu_k \in \mathcal{U}([-1/2 \text{ LSB}, +1/2 \text{ LSB}]) \\ &\text{and } Y_{\nu\nu} \approx Y_{\text{rnd}} \\ &\qquad = 12/\text{LSB}^2. \end{aligned}$$

The analog of the χ^2 statistic in this case is the information-normalized square of the innovations $\nu = \varepsilon_{\text{rnd}}$

$$\begin{aligned} |\varepsilon_{\text{rnd}}|_Y^2 &\stackrel{\text{def}}{=} \varepsilon_{\text{rnd}}^2 \times Y_{\text{rnd}} \\ &= \varepsilon_{\text{rnd}}^2 \frac{12}{\text{LSB}^2}, \end{aligned}$$

the mean of which is 1 (one), the same as the mean of the χ^2 distribution with one degree of freedom, χ_1^2. Therefore, sensor roundoff errors—even though they are uniformly distributed—will pass the zero-mean and chi-squared mean tests. However, the distribution of this statistic is not chi-squared.

Using Equation 3.127, the distribution of $y = |\varepsilon_{\text{rnd}}|_Y^2$ can be derived as

$$p(y) = \begin{cases} 0, & y < 0 \\ \frac{1}{2\sqrt{3y}}, & 0 \le y \le 3, \\ 0, & y > 3 \end{cases}$$

as plotted in Figure 9.1. Figure 9.1(a) shows the roundoff error distribution compared to the Gaussian distribution with the same mean and variance. Figure 9.1 (b) is a semi-log plot showing the relevant squared roundoff distribution compared to the χ_1^2 distribution.

These results would indicate that an empirical distribution based on observed innovations may be preferable to the chi-square distribution for assessing whether an observed innovation is acceptable.

9.2.1.8 Real-Time Monitoring Because numerical values of the innovations ν_k and their alleged information matrix $Y_{\nu\nu k}$ are available "for free" within the Kalman filter implementation, the marginal computational cost of monitoring innovations statistics is relatively low. The expected payoffs for such monitoring will depend on attributes of the application. In all cases, however, such monitoring will generally require

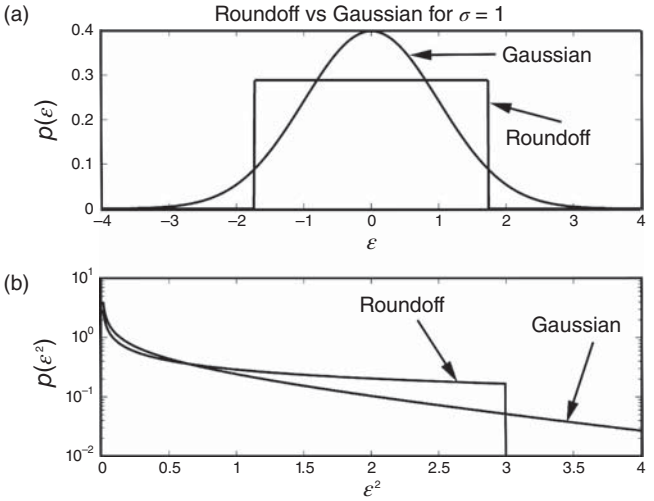

Figure 9.1 Distributions of roundoff errors.

1. Selection of which statistics to monitor, based on relative efficacy in separating anomalous events from normal events.
2. Determination of threshold levels for determining which events are out of tolerance. This may involve some analysis related to the relative costs of false negatives and false positives.
3. Exception-handling software for making decisions about what needs to be done and carrying out the necessary actions.

It is common practice in some applications to save the monitored statistics for offline analysis, as well—particularly during early testing and evaluation.

Monitoring of this sort may also be done for spotting performance degradation trends, based on long-term statistical values.

9.2.2 Convergence and Divergence

9.2.2.1 Some Definitions A sequence $\{\eta_k | k = 1, 2, 3, \ldots\}$ of real vectors η_k is said to *converge* to a *limit* η_∞ if, for every $\varepsilon > 0$, for some n, for all $k > n$, the norm of the differences $\|\eta_k - \eta_\infty\| < \varepsilon$. Let us use the expressions

$$\lim_{k\to\infty} \eta_k = \eta_\infty \text{ or } \eta_k \to \eta_\infty$$

to represent convergence. One vector sequence is said to converge to another vector sequence if their differences converge to the zero vector, and a sequence is said to converge[3] if, for every $\varepsilon > 0$, for some integer n, for all $k, \ell > n$, $\|\eta_k - \eta_\ell\| < \varepsilon$.

[3]Such sequences are called *Cauchy sequences*, after Augustin Louis Cauchy (1789–1857).

Divergence is defined as convergence to ∞: for every $\varepsilon > 0$, for some integer n, for all $k > n$, $|\eta_k| > \varepsilon$. In that case, $\|\eta_k\|$ is said to *grow without bound*.

9.2.2.2 Nonconvergence This is a more common issue in the performance of Kalman filters than strict divergence. That is, the filter fails because it does not converge to the desired limit, although it does not necessarily diverge.

9.2.2.3 Variables Subject to Divergence The operation of a Kalman filter involves the following sequences that may or may not converge or diverge:

x_k, the sequence of actual state values;
$\mathrm{E}\langle x_k x_k^{\mathrm{T}}\rangle$, the mean-squared state;
$\hat{x}_k$, the estimated state;
$\tilde{x}_{k(-)} = \hat{x}_{k(-)} - x_k$, the a priori estimation error;
$\tilde{x}_{k(+)} = \hat{x}_{k(+)} - x_k$, the a posteriori estimation error;
$P_{k(-)}$, the covariance of a priori estimation errors;
$P_{k(+)}$, the covariance of a posteriori estimation errors.

One may also be interested in whether or not the sequences $\{P_{k(-)}\}$ and $\{P_{k(+)}\}$ computed from the Riccati equations converge to the corresponding *true* covariances of estimation error.

9.2.3 Covariance Analysis

The covariance matrix of estimation uncertainty characterizes the theoretical performance of the Kalman filter. It is computed as an ancillary variable in the Kalman filter as the solution of a matrix Riccati equation with the given initial conditions. It is also useful for predicting performance. If its characteristic values are growing without bound, then the theoretical performance of the Kalman filter is said to be diverging. This can happen if the system state is unstable and unobservable, for example. This type of divergence is detectable by solving the Riccati equation to compute the covariance matrix.

The Riccati equation is not always well conditioned for numerical solution and one may need to use the more numerically stable methods of Chapter 7 to obtain reasonable results. One can, for example, use eigenvalue–eigenvector decomposition of solutions to test their characteristic roots (they should be positive) and condition numbers. Condition numbers within one or two orders of magnitude of ε^{-1} (the reciprocal of the unit roundoff error in computer precision) are considered probable cause for concern and reason to use square-root methods.

9.2.4 Testing for Unpredictable Behavior

Not all filter divergence is predictable from the Riccati equation solution. Sometimes the actual performance does not agree with theoretical performance.

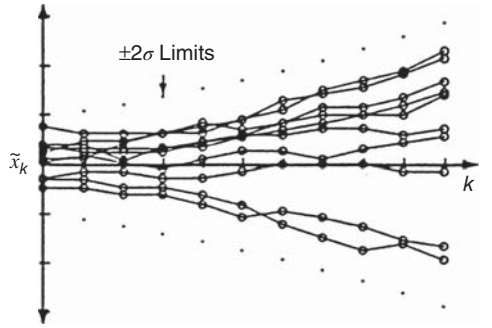

Figure 9.2 Dispersion of multiple runs.

One cannot measure estimation error directly, except in simulations, so one must find other means to check on estimation accuracy. Whenever the estimation error is deemed to differ significantly from its expected value (as computed by the Riccati equation), the filter is said to diverge from its predicted performance. We will now consider how one might go about detecting divergence.

Examples of typical behaviors of Kalman filters are shown in Figure 9.2, which is a multiplot of the estimation errors on 10 different simulations of a filter implementation with independent pseudorandom-error sequences. Note that each time the filter is run, different estimation errors $\tilde{x}(t)$ result, even with the same initial condition $\hat{x}(0)$. Also note that at any particular time the average estimation error (across the ensemble of simulations) is approximately zero,

$$\frac{1}{N}\sum_{i=1}^{N}[\hat{x}_{[i]}(t_k) - x(t_k)] \approx \mathop{\mathrm{E}}_{i}\langle \hat{x}_{[i]}(t_k) - x(t_k)\rangle = 0, \tag{9.11}$$

where N is the number of simulation runs and $\hat{x}_{[i]}(t_k) - x(t_k)$ is the estimation error at time t_k on the ith simulation run.

Monte Carlo analysis of Kalman filter performance uses many such runs to test that the ensemble mean estimation error is *unbiased* (i.e., has effectively zero mean) and that its ensemble covariance is in close agreement with the theoretical value computed as a solution of the Riccati equation.

9.2.4.1 Convergence of Suboptimal Filters In the suboptimal filters discussed in Section 9.5, the estimates can be biased. Therefore, in the analysis of suboptimal filters, the behavior of $P(t)$ is not sufficient to define convergence. A suboptimal filter is said to converge if the covariance matrices converge,

$$\lim_{t\to\infty}[\mathrm{trace}(P_{\text{sub-opt}} - P_{\text{opt}})] = 0, \tag{9.12}$$

and the asymptotic estimation error is unbiased,

$$\lim_{t\to\infty}\mathrm{E}[\tilde{x}(t)] = 0. \tag{9.13}$$

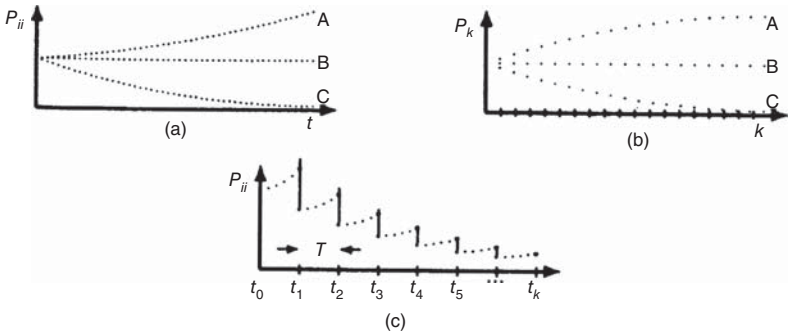

Figure 9.3 Asymptotic behaviors of estimation uncertainties. (a) Continuous time, (b) discrete time, and (c) discrete measurement with continuous dynamics.

Example 9.3 Some typical behavior patterns of suboptimal filter convergence are depicted by the plots of $P(t)$ in Figure 9.3(a), and characteristics of systems with these symptoms are given here as examples.

Case A: Let a scalar continuous system equation be given by

$$\dot{x}(t) = Fx(t), \qquad F > 0, \tag{9.14}$$

in which the system is unstable, or

$$\dot{x}(t) = Fx(t) + w(t) \tag{9.15}$$

in which the system has driving noise and is unstable.

Case B: The system has constant steady-state uncertainty:

$$\lim_{t\to\infty} \dot{P}(t) = 0. \tag{9.16}$$

Case C: The system is stable and has no driving noise:

$$\dot{x}(t) = -Fx(t), \qquad F > 0. \tag{9.17}$$

Example 9.4 (Behaviors of Discrete-Time Systems) Plots of P_k are shown in Figure 9.3(b) for the following system characteristics:

Case A: Effects of dynamic disturbance noise and measurement noise are large relative to $P_0(t)$ (initial uncertainty).

Case B: $P_0 = P_\infty$ (Wiener filter).

Case C: Effects of dynamic disturbance noise and measurement noise are small relative to $P_0(t)$.

Example 9.5 (Continuous System with Discrete Measurements) A scalar example of a behavior pattern of the covariance propagation equation ($P_k(-), \dot{P}(t)$) and covariance update equation $P_k(+)$,

$$\dot{x}(t) = Fx(t) + w(t), F < 0,$$
$$z(t) = x(t) + v(t),$$

is shown in Figure 9.3(c).

The following features may be observed in the behavior of $P(t)$:

1. Processing the measurement tends to reduce P.
2. Process noise covariance (Q) tends to increase P.
3. Damping in a stable system tends to reduce P.
4. Unstable system dynamics ($F > 0$) tend to increase P.
5. With white Gaussian measurement noise, the time between samples (T) can be reduced to decrease P.

The behavior of P represents a composite of all these effects (1–5) as shown in Figure 9.3(c).

9.2.4.2 Causes of Predicted Nonconvergence Nonconvergence of P predicted by the Riccati equation can be caused by

1. "natural behavior" of the dynamic equations or
2. nonobservability with the given measurements.

The following examples illustrate these behavioral patterns.

Example 9.6 The "natural behavior" for P in some cases is for

$$\lim_{t\to\infty} P(t) = P_\infty \text{ (a constant).} \tag{9.18}$$

For example,

$$\left.\begin{array}{ll} \dot{x} = w, & \text{cov}(w) = Q \\ z = x + v, & \text{cov}(v) = R \end{array}\right) \Rightarrow \begin{array}{l} F = 0 \\ G = H = 1 \end{array} \quad \text{in} \quad \begin{array}{l} \dot{x} = Fx + Gw, \\ z = Hx + v. \end{array} \tag{9.19}$$

Applying the continuous Kalman filter equations from Chapter 5, then

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - \overline{K}R\overline{K}^{\mathrm{T}}$$

and

$$\overline{K} = PH^{\mathrm{T}}R^{-1}$$

become

$$\dot{P} = Q - \overline{K}^2 R$$

and

$$\overline{K} = \frac{P}{R}$$

or

$$\dot{P} = Q - \frac{P^2}{R}.$$

The solution is

$$P(t) = \alpha \left(\frac{P_0 \cosh(\beta t) + \alpha \sinh(\beta t)}{P_0 \sinh(\beta t) + \alpha \cosh(\beta t)} \right), \tag{9.20}$$

where

$$\alpha = \sqrt{RQ}, \quad \beta = \sqrt{Q/R}. \tag{9.21}$$

Note that the solution of the Riccati equation converges to a finite limit:

1. $\lim_{t\to\infty} P(t) = \alpha > 0$, a finite, but nonzero, limit. (See Figure 9.4(a).)
2. This is no cause for alarm, and there is no need to remedy the situation if the asymptotic mean-squared uncertainty is tolerable. If it is not tolerable, then the remedy must be found in the hardware (e.g., by attention to the physical sources of R or Q—or both) and not in software.

Example 9.7 (Divergence Due to "Structural" Unobservability) The filter is said to diverge at infinity if its limit is unbounded:

$$\lim_{t\to\infty} P(t) = \infty. \tag{9.22}$$

As an example in which this occurs, consider the system

$$\begin{aligned} \dot{x}_1 &= w, & \operatorname{cov}(w) &= Q, \\ \dot{x}_2 &= 0, & \operatorname{cov}(v) &= R, \\ z &= x_2 + v, \end{aligned} \tag{9.23}$$

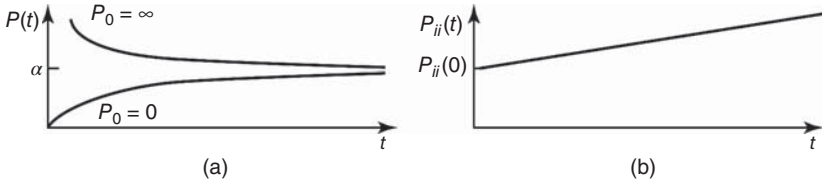

Figure 9.4 Behavior patterns of P. (a) Convergent to finite limit and (b) convergent to infinite limit.

with initial conditions

$$P_0 = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} P_{11}(0) & 0 \\ 0 & P_{22}(0) \end{bmatrix}. \tag{9.24}$$

The continuous Kalman filter equations

$$\begin{aligned} \dot{P} &= FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - \overline{K}R\overline{K}^{\mathrm{T}}, \\ \overline{K} &= PH^{\mathrm{T}}R^{-1} \end{aligned}$$

can be combined to give

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP, \tag{9.25}$$

or

$$\dot{P}_{11} = Q - \frac{p_{12}^2}{R}, \quad \dot{P}_{12} = -\frac{p_{12}p_{22}}{R}, \quad \dot{P}_{22} = -\frac{p_{22}^2}{R}, \tag{9.26}$$

the solution to which is

$$p_{11}(t) = p_{11}(0) + Qt, \quad p_{12}(t) = 0, \quad p_{22}(t) = \frac{p_{22}(0)}{1 + [p_{22}(0)/R]t}, \tag{9.27}$$

as plotted in Figure 9.4(b). The only remedy in this example is to alter or add measurements (sensors) to achieve observability.

Example 9.8 (Nonconvergence Due to "Structural" Unobservability) Parameter estimation problems have no state dynamics and no process noise. One might reasonably expect the estimation uncertainty to approach zero asymptotically as more and more measurements are made. However, it can still happen that the filter will not converge to absolute certainty. That is, the asymptotic limit of the estimation uncertainty

$$0 < \lim_{k \to \infty} P_k < \infty \tag{9.28}$$

is actually bounded away from zero uncertainty.

Parameter Estimation Model for Continuous Time Consider the two-dimensional parameter estimation problem

$$\left.\begin{aligned} \dot{x}_1 = 0, \qquad \dot{x}_2 = 0, P_0 = \begin{bmatrix} \sigma_1^2(0) & 0 \\ 0 & \sigma_2^2(0) \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 1 \end{bmatrix}, \\ z = H \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \qquad \mathrm{cov}(v) = R, \end{aligned}\right\} \tag{9.29}$$

in which only the *sum* of the two state variables is measurable. The difference of the two state variables will then be unobservable.

Problem in Discrete Time This example also illustrates a difficulty with a standard shorthand notation for discrete-time dynamic systems: the practice of using subscripts to indicate discrete time. Subscripts are more commonly used to indicate components of vectors. The solution here is to move the component indices "upstairs" and make them superscripts. (This approach only works here because the problem is linear. Therefore, one does not need superscripts to indicate powers of the components.) For these purposes, let x_k^i denote the ith component of the state vector at time t_k. The continuous form of the parameter estimation problem can then be "discretized" to a model for a discrete Kalman filter (for which the state-transition matrix is the identity matrix; see Section 5.2):

$$x_k^1 = x_{k-1}^1 \quad (x^1 \text{ is constant}), \tag{9.30}$$

$$x_k^2 = x_{k-1}^2 \quad (x^2 \text{ is constant}), \tag{9.31}$$

$$z_k = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} + v_k. \tag{9.32}$$

Let

$$\hat{x}_0 = 0.$$

The estimator then has two sources of information from which to form an optimal estimate of x_k:

1. the a priori information in $\hat{x}_0$ and P_0 and
2. the measurement sequence $z_k = x_k^1 + x_k^2 + v_k$ for $k = 1, 2, 3, \ldots$

In this case, the best the optimal filter can do with the measurements is to "average out" the effects of the noise sequence $v_1, \ldots, v_k$. One might expect that an infinite number of measurements (z_k) would be equivalent to *one* noise-free measurement, that is,

$$z_1 = (x_1^1 + x_1^2), \quad \text{where } v_1 \to 0 \quad \text{and} \quad R = \text{cov}(v_1) \to 0. \tag{9.33}$$

Estimation Uncertainty from a Single Noise-Free Measurement By using the discrete filter equations with one stage of estimation on the measurement z_1, one can obtain the gain in the form

$$\overline{K}_1 = \begin{bmatrix} \dfrac{\sigma_1^2(0)}{(\sigma_1^2(0) + \sigma_2^2(0) + R)} \\ \dfrac{\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \end{bmatrix}. \tag{9.34}$$

The estimation uncertainty covariance matrix can then be shown to be

$$P_{1(+)} = \begin{bmatrix} \dfrac{\sigma_1^2(0)\sigma_2^2(0) + R\sigma_1^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} & \dfrac{-\sigma_1^2(0)\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \\ \dfrac{-\sigma_1^2(0)\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} & \dfrac{\sigma_1^2(0)\sigma_2^2(0) + R\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0) + R} \end{bmatrix} \equiv \begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix}, \tag{9.35}$$

where the *correlation coefficient* (defined in Equation 3.40) is

$$\rho_{12} = \frac{p_{12}}{\sqrt{p_{11}p_{22}}} = \frac{-\sigma_1^2(0)\sigma_2^2(0)}{\sqrt{[\sigma_1^2(0)\sigma_2^2(0) + R\sigma_1^2(0)][\sigma_1^2(0)\sigma_2^2(0) + R\sigma_2^2(0)]}}, \tag{9.36}$$

and the state estimate is

$$\hat{x}_1 = \hat{x}_1(0) + \overline{K}_1[z_1 - H\hat{x}_1(0)] = [I - \overline{K}_1\, H]\hat{x}_1(0) + \overline{K}_1 z_1. \tag{9.37}$$

However, for the *noise-free* case,

$$v_1 = 0, \quad R = \text{cov}(v_1) = 0,$$

the correlation coefficient is

$$\rho_{12} = -1, \tag{9.38}$$

and the estimates for $\hat{x}_1(0) = 0$,

$$\hat{x}_1^1 = \left(\frac{\sigma_1^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)} \right)(x_1^1 + x_1^2),$$
$$\hat{x}_1^2 = \left(\frac{\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)} \right)(x_1^1 + x_1^2),$$

are totally insensitive to the difference $x_1^1 - x_1^2$. As a consequence, the filter will *almost never get the right answer*! This is a fundamental characteristic of the problem, however, and not attributable to the design of the filter. There are *two* unknowns (x_1^1 and x_1^2) and *one* constraint:

$$z_1 = (x_1^1 + x_1^2). \tag{9.39}$$

Conditions for Serendipitous Design The conditions under which the filter will still get the right answer can easily be derived. Because x_1^1 and x_1^2 are constants, their ratio constant

$$C \stackrel{\text{def}}{=} \frac{x_1^2}{x_1^1} \tag{9.40}$$

will also be a constant, such that the sum

$$\begin{aligned} x_1^1 + x_1^2 &= (1 + C)x_1^1 \\ &= \left(\frac{1 + C}{C}\right) x_1^2. \end{aligned}$$

Then

$$\hat{x}_1^1 = \left(\frac{\sigma_1^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)}\right)[(1 + C)x_1^1] = x_1^1 \quad \text{only if} \quad \frac{\sigma_1^2(0)(1 + C)}{\sigma_1^2(0) + \sigma_2^2(0)} = 1,$$

$$\hat{x}_1^2 = \left(\frac{\sigma_2^2(0)}{\sigma_1^2(0) + \sigma_2^2(0)}\right)\left(\frac{1 + C}{C}\right) x_1^2 = x_1^2 \quad \text{only if} \quad \frac{\sigma_2^2(0)(1 + C)}{[\sigma_1^2(0) + \sigma_2^2(0)](C)} = 1.$$

Both these conditions are satisfied *only if*

$$\frac{\sigma_2^2(0)}{\sigma_1^2(0)} = C = \frac{x_1^2}{x_1^1} \geq 0, \tag{9.41}$$

because $\sigma_1^2(0)$ and $\sigma_2^2(0)$ are nonnegative numbers.

Likelihood of Serendipitous Design For the filter to obtain the right answer, it would be necessary that

1. x_1^1 and x_1^2 have the *same sign* and
2. it is known that their ratio $C = x_1^2/x_1^1$.

Since both of these conditions are rarely satisfied, the filter estimates would rarely be correct.

What Can Be Done About It? The following methods can be used to detect nonconvergence due to this type of structural unobservability:

- Test the system for observability using the "observability theorems" of Section 2.5.
- Look for perfect correlation coefficients ($\rho = \pm 1$) and be very suspicious of high correlation coefficients (e.g., $|\rho| > 0.9$).

- Perform eigenvalue–eigenvector decomposition of P to test for negative characteristic values or a large condition number. (This is a better test than correlation coefficients to detect unobservability.)
- Test the filter on a system simulator with *noise-free outputs* (measurements).

Remedies for this problem include

- attaining observability by adding another type of measurement or
- defining $\acute{x} \equiv x^1 + x^2$ as the only state variable to be estimated.

Example 9.9 (Unobservability Caused by Poor Choice of Sampling Rate) The problem in Example 9.8 might be solved by using an additional measurement—or by using a measurement with a time-varying sensitivity matrix. Next, consider what can go wrong even with time-varying measurement sensitivities if the sampling rate is chosen badly. For that purpose, consider the problem of estimating unknown parameters (constant states) with both constant and sinusoidal measurement sensitivity matrix components:

$$H(t) = [1 \quad \cos(\omega t)],$$

as plotted in Figure 9.5. The equivalent model for use in the discrete Kalman filter is

$$x_k^1 = x_{k-1}^1, \quad x_k^2 = x_{k-1}^2, \quad H_k = H(kT), \quad z_k = H_k \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} + v_k,$$

where T is the intersample interval.

What Happens When Murphy's Law Takes Effect? With the choice of intersampling interval as $T = 2\pi/\omega$ and $t_k = kT$, the components of the measurement sensitivity matrix become equal and constant:

$$\begin{aligned} H_k &= [1 \quad \cos(\omega kT)] \\ &= [1 \quad \cos(2\pi k)] \\ &= [1 \quad 1], \end{aligned}$$

as shown in Figure 9.5. (This is the way many engineers discover "aliasing.") The states x^1 and x^2 are *unobservable* with this choice of sampling interval (see

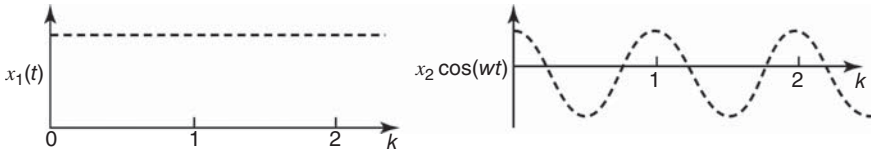

Figure 9.5 Aliased measurement components.

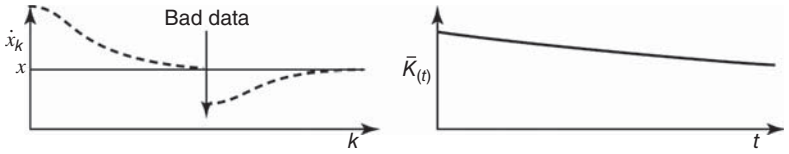

Figure 9.6 Asymptotic recovery from bad data.

Figure 9.5). With this choice of sampling times, the system and filter behave as in the previous example.

Methods for detecting and correcting unobservability include those given in Example 9.8 plus the more obvious remedy of changing the sampling interval T to obtain observability, for example,

$$T = \frac{\pi}{\omega} \tag{9.42}$$

is a better choice.

Causes of Unpredicted Nonconvergence Unpredictable nonconvergence may be caused by

1. bad data,
2. numerical problems, or
3. mismodeling.

Example 9.10 (Unpredicted Nonconvergence Due to Bad Data) "Bad data" are caused by something going wrong, which is almost sure to happen in real-world applications of Kalman filtering. These verifications of Murphy's law occur principally in two forms:

- The initial estimate is badly chosen, for example,

$$|\hat{x}(0) - x|^2 = |\tilde{x}|^2 \gg \text{trace} P_0. \tag{9.43}$$

- The measurement has an exogenous component (a mistake, not an error) that is excessively large, for example,

$$|v|^2 \gg \text{trace} R. \tag{9.44}$$

Asymptotic Recovery from Bad Data In either case, if the system is truly linear, the Kalman filter will (in theory) recover in finite time as it continues to use measurements z_k to estimate the state x. (The best way is to prevent bad data from getting into the filter in the first place!) See Figure 9.6.

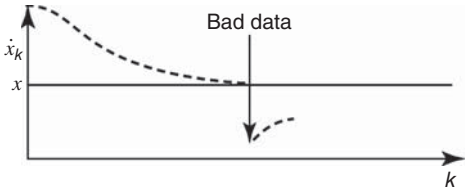

Figure 9.7 Failure to recover in short period.

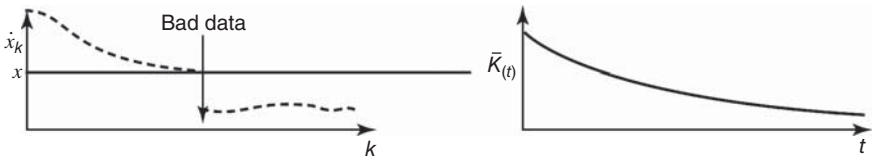

Figure 9.8 Failure to recover due to gain decay.

Practical Limitations of Recovery Often, in practice, the recovery is not adequate in finite time. The interval $(0, T)$ of measurement availability is fixed and may be too short to allow sufficient recovery (see Figure 9.7). The normal behavior of the gain matrix $\overline{K}$ may be too rapidly converging toward its steady-state value of $\overline{K} = 0$. (See Figure 9.8.)

9.2.4.3 Heading Off Bad Data

- Inspection of $P(t)$ and $\overline{K}(t)$ is useless, because they are not affected by data.
- Inspection of the state estimates $\hat{x}(t)$ for sudden jumps (*after* a bad measurement has already been used by the filter) is sometimes used, but it still leaves the problem of undoing the damage to the estimate after it has been done.
- Inspection of the "innovations" vector $[z - H\hat{x}]$ for sudden jumps or large entries (*before* bad measurement is processed by the filter) is much more useful, because the discrepancies can be interpreted probabilistically, and the data can be discarded before it has spoiled the estimate (see Section 9.3).

The best remedy for this problem is to implement a "bad data detector" to reject the bad data before it contaminates the estimate. If this is to be done in real time, it is sometimes useful to *save* the bad data for offline examination by an "exception handler" (often a human, but sometimes a second-level data analysis program) to locate and remedy the causes of the bad data that are occurring.

Artificially Increasing Process Noise Covariance to Improve Bad Data Recovery If bad data are detected after they have been used, one can keep the filter "alive" (to pay more attention to subsequent data) by increasing the process noise covariance Q in the system model assumed by the filter. Ideally, the new process noise covariance

should reflect the actual measurement error covariance, including the bad data as well as other random noise.

Example 9.11 (Nonconvergence Due to Numerical Problems) This is sometimes detected by observing impossible P_k behavior. The terms on the main diagonal of P_k may become negative, or *larger*, immediately after a measurement is processed than immediately before, that is, $\sigma(+) > \sigma(-)$. A less obvious (but detectable) failure mode is for the *characteristic values* of P to become negative. This can be detected by eigenvalue–eigenvector decomposition of P. Other means of detection include using simulation (with known states) to compare estimation errors with their estimated covariances. One can also use double precision in place of single precision to detect differences due to precision. Causes of numerical problems can sometimes be traced to inadequate wordlength (precision) of the host computer. These problems tend to become worse with larger numbers of state variables.

Remedies for Numerical Problems These problems have been treated by many "brute-force" methods, such as using higher precision (e.g., double instead of single). One can try reducing the number of states by merging or eliminating unobservable states, eliminating states representing "small effects," or using other suboptimal filter techniques such as decomposition into lower dimensional state spaces.

Possible remedies include the use of *more numerically stable methods* (to obtain more computational accuracy with the same computer precision) and the use of *higher precision*. The latter approach (higher precision) will increase the execution time but will generally require less reprogramming effort. One can sometimes use a better algorithm to improve the accuracy of matrix inverse $(HPH^{\mathrm{T}} + R)^{-1}$ (e.g., the Cholesky decomposition method shown in Chapter 7) or eliminate the inverse altogether by diagonalizing R and processing the measurements sequentially (also shown in Chapter 7).

9.2.5 Effects Due to Mismodeling

9.2.5.1 Lying to the Filter The Kalman gain and the covariance matrix P are correct *if* the models used in computing them are correct. With mismodeling, the P matrix can be erroneous and of little use in detecting nonconvergence, or P can even converge to zero while the state estimation error $\tilde{x}$ is actually diverging. (It happens.)

The problems that result from bad initial estimates of x or P have already been addressed. There are four other types that will now be addressed:

1. unmodeled state variables of the dynamic system,
2. unmodeled process noise,
3. errors in the dynamic coefficients or state-transition matrix, and
4. overlooked nonlinearities.

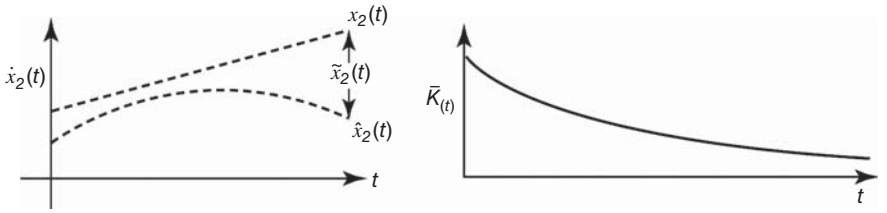

Figure 9.9 Divergence due to mismodeling.

Example 9.12 (Nonconvergence Caused by Unmodeled State Variables) Consider the following example:

Real-World (Creeping State)	Kalman Filter Model (Constant State)
$\dot{x}_1 = 0$	$\dot{x}_2 = 0$
$\dot{x}_2 = x_1$	$z = x_2 + v$
$z = x_2 + v$	
$\Rightarrow x_2(t) = x_2(0) + x_1(0)t$	$\Rightarrow x_2(t) = x_2(0)$

(9.45)

for which, in the filter model, the Kalman gain $\overline{K}(t) \to 0$ as $t \to \infty$. The filter is unable to provide feedback for the error in the estimate of $x_2(t)$ as it grows in time (even if it grows slowly). Eventually $\tilde{x}_2(t) = \hat{x}_2(t) - x_2(t)$ diverges as shown in Figure 9.9.

Detecting Unmodeled State Dynamics by Fourier Analysis of the Filter Innovations It is difficult to diagnose unmodeled state variables unless all other causes for nonconvergence have been ruled out. If there is high confidence in the model being used, then simulation can be used to rule out any of the other causes above. Once these other causes have been eliminated, Fourier analysis of the filter innovations (the prediction errors $\{z_k - H_k\hat{x}_k\}$) can be useful for spotting characteristic frequencies of the unmodeled state dynamics. If the filter is modeled properly, then the innovations should be uncorrelated, having a power spectral density (PSD) that is essentially flat. Persistent peaks in the PSD would indicate the characteristic frequencies of unmodeled effects. These peaks can be at zero frequency, which would indicate a bias error in the innovations.

Remedies for Unmodeled State Variables The best cure for nonconvergence caused by unmodeled states is to correct the model, but this not always easy to do. As an ad hoc fix, additional "fictitious" process noise can be added to the system model assumed by the Kalman filter.

Example 9.13 (Adding "Fictitious" Process Noise to the Kalman Filter Model) Continuing with the continuous-time problem of Example 7.10, consider the alternative Kalman filter model

$$\dot{x}_2(t) = w(t), \quad z(t) = x_2(t) + v(t).$$

"Type-1 Servo" Behavior The behavior of this filter can be analyzed by applying the continuous Kalman filter equations from Chapter 5, with parameters

$$F = 0, \quad H = 1, \quad G = 1,$$

transforming the general Riccati differential equation

$$\begin{aligned}\dot{P} &= FP + PF^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP + GQG^{\mathrm{T}} \\ &= \frac{-P^2}{R} + Q\end{aligned}$$

to a scalar equation with steady-state solution (to $\dot{P} = 0$)

$$P(\infty) = \sqrt{RQ}.$$

The steady-state Kalman gain

$$\overline{K}(\infty) = \frac{P(\infty)}{R} = \sqrt{\frac{Q}{R}}. \tag{9.46}$$

The equivalent steady-state model of the Kalman filter can now be formulated as follows:[2]

$$\dot{\hat{x}}_2 = F\hat{x}_2 + \overline{K}[z - H\hat{x}_2]. \tag{9.47}$$

Here, $F = 0, H = 1, \overline{K} = \overline{K}(\infty), \hat{x} = \hat{x}_2$, so that

$$\dot{\hat{x}}_2 + \overline{K}(\infty)\hat{x}_2 = \overline{K}(\infty)z. \tag{9.48}$$

The steady-state response of this estimator can be determined analytically by taking Laplace transforms:

$$[s + \overline{K}(\infty)]\hat{x}_2(s) = \overline{K}(\infty)z(s) \tag{9.49}$$

$$\Rightarrow \frac{\hat{x}_2(s)}{z(s)} = \frac{\overline{K}(\infty)}{s + \overline{K}(\infty)}. \tag{9.50}$$

Figure 9.10(a) shows a "type-1 servo". Its steady-state following error (even in the noise-free case) is not zero in the real-world case:

$$z(t) = x_2(t) = x_2(0) + x_1(0)t \quad \text{with } v = 0.$$

Taking the Laplace transform of the equation yields

$$z(s) = x_2(s) = \frac{x_2(0)}{s} + \frac{x_1(0)}{s^2}.$$

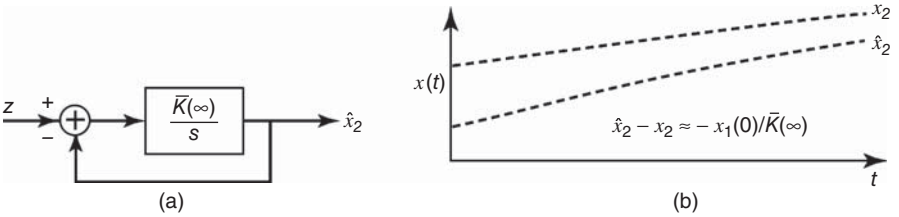

Figure 9.10 Type-1 servo (a) and estimates (b).

The error in $x_2(t)$ is

$$\tilde{x}_2(t) = \hat{x}_2(t) - x_2(t).$$

Taking the Laplace of the equation and substituting the value of $\hat{x}_2(s)$ from Equation 9.50 give

$$\begin{aligned}\tilde{x}_2(s) &= \frac{\overline{K}(\infty)}{s + \overline{K}(\infty)} x_2(s) - x_2(s) \\ &= \left[-\frac{s}{s + \overline{K}(\infty)} \right] x_2(s).\end{aligned}$$

Applying the final-value theorem, one gets

$$\begin{aligned}\tilde{x}_2(\infty) &= [\hat{x}_2(\infty) - x_2(\infty)] = \lim_{s \to 0} s[\hat{x}_2(s) - x_2(s)] \\ &= \lim_{s \to 0} s \left[-\frac{s}{s + \overline{K}(\infty)} \right] [x_2(s)] \\ &= \lim_{s \to 0} s \left[-\frac{s}{s + \overline{K}(\infty)} \right] \left[\frac{x_2(0)}{s} + \frac{x_1(0)}{s^2} \right] \\ &= -\frac{x_1(0)}{\overline{K}(\infty)} (\text{a bias}).\end{aligned}$$

This type of behavior is shown in Figure 9.10(b).

If the steady-state bias in the estimation error is unsatisfactory with the approach in Example 7.11, one can go one step further by adding another state variable and fictitious process noise to the system model assumed by the Kalman filter.

Example 9.14 (Effects of Adding States and Process Noise to the Kalman Filter Model) Suppose that the model of Example 7.11 was modified to the following form:

Real World	Kalman Filter Model
$\dot{x}_1 = 0$	$\dot{x}_1 = w$
$\dot{x}_2 = x_1$	$\dot{x}_2 = x_1$
$z = x_2 + v$	$z = x_2 + v$

(9.51)

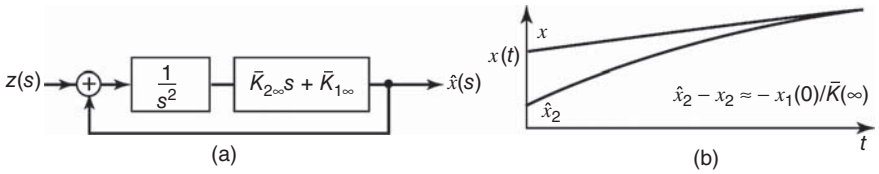

Figure 9.11 Type-2 servo (a) and servo estimates (b).

That is, $x_2(t)$ is now modeled as an "integrated random walk." In this case, the steady-state Kalman filter has an additional integrator and behaves like a "type-2 servo" (see Figure 9.11(a)). The type-2 servo has zero steady-state following error to a ramp. However, its transient response may become more sluggish and its steady-state error due to noise is not zero, the way it would be with the real world correctly modeled. Here,

$$F = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad H = [0 \quad 1], \quad Q = \text{cov}(w), \quad R = \text{cov}(v) \tag{9.52}$$

$$\dot{P} = FP + PF^{\mathrm{T}} + GQG^{\mathrm{T}} - PH^{\mathrm{T}}R^{-1}HP \text{ and } \overline{K} = PH^{\mathrm{T}}R^{-1} \tag{9.53}$$

become in the steady state

$$\left.\begin{aligned} \dot{P}_{11} &= Q - \frac{p_{12}^2}{R} = 0 \\ \dot{P}_{12} &= p_{11} - \frac{p_{12}p_{22}}{R} = 0 \\ \dot{P}_{22} &= 2p_{12} - \frac{p_{22}^2}{R} = 0 \end{aligned}\right\} \Rightarrow \begin{aligned} p_{12}(\infty) &= \sqrt{RQ}, \\ p_{22}(\infty) &= \sqrt{2}(R^3Q)^{1/4}, \\ p_{11}(\infty) &= \sqrt{2}(Q^3R)^{1/4}, \\ \overline{K}(\infty) &= \begin{bmatrix} \sqrt{\frac{Q}{R}} \\ \sqrt{2}\sqrt[4]{\frac{Q}{R}} \end{bmatrix} = \begin{bmatrix} \overline{K}_1(\infty) \\ \overline{K}_2(\infty) \end{bmatrix}, \end{aligned} \tag{9.54}$$

and these can be Laplace transformed to yield

$$\hat{x}(s) = [sI - F + \overline{K}(\infty)H]^{-1}\overline{K}(\infty)z(s). \tag{9.55}$$

In component form, this becomes

$$\hat{x}_2(s) = \frac{(\overline{K}_2 s + \overline{K}_1)/s^2}{1 + (\overline{K}_2 s + \overline{K}_1)/s^2} z(s). \tag{9.56}$$

The resulting steady-state following error to a ramp (in the noise-free case) is easily determined:

$$z(t) = x_2(t) = x_2(0) + x_1(0)t, \quad v = 0,$$

$$\tilde{x}_2(s) = \hat{x}_2(s) - x_2(s) = -\left(\frac{s^2}{s^2 + \overline{K}_2 s + \overline{K}_1}\right) x_2(s),$$

$$\tilde{x}_2(\infty) = \lim_{s\to 0} s\left(\frac{-s^2}{s^2 + \overline{K}_2 s + \overline{K}_1}\right)\left(\frac{x_2(0)}{s} + \frac{x_1(0)}{s^2}\right) = 0.$$

This type of behavior is shown in Figure 9.11(b).

Example 9.15 (Statistical Modeling Errors Due to Unmodeled Dynamic Distubrance Noise) With the models

$$\begin{array}{cc} \underline{\text{Real World}} & \underline{\text{Kalman Filter Model}} \\ \dot{x}_1 = w & \dot{x}_1 = 0 \\ \dot{x}_2 = x_1 & \dot{x}_2 = x_1 \\ z = x_2 + v & z = x_2 + v \end{array} \tag{9.57}$$

the Kalman filter equations yield the following relationships:

$$\left.\begin{array}{l} \dot{p}_{11} = \frac{-1}{R} p_{12}^2 \\ \dot{p}_{12} = p_{11} - \dfrac{p_{12}p_{22}}{R} \\ \dot{p}_{22} = 2p_{12} - \frac{p_{22}^2}{R} \end{array}\right\} \quad \Rightarrow \quad \text{in the steady state:} \quad \begin{array}{l} p_{12} = 0, \quad \overline{K} = PH^{\mathrm{T}}R^{-1} = 0, \\ p_{11} = 0, \quad \hat{x}_1 = \text{const}, \\ p_{22} = 0, \quad \hat{x}_2 = \text{ramp}. \end{array} \tag{9.58}$$

Since x_1 is not constant (due to the driving noise w), the state estimates will not converge to the states, as illustrated in the simulated case plotted in Figure 9.12. This figure shows the behavior of the Kalman gain $\overline{K}_1$, the estimated state component $\hat{x}_1$, and the "true" state component x_1 in a discrete-time simulation of the above model. Because the assumed process noise is zero, the Kalman filter gain $\overline{K}_1$ converges to zero. Because the gain converges to zero, the filter is unable to track the errant state vector component x_1, a random-walk process. Because the filter is unable to track the true state, the innovations (the difference between the predicted measurement and the actual measurement) continue to grow without bound. Even though the gains are converging to zero, the product of the gain and the innovations (used in updating the state estimate) can be significant.

In this particular example, $\sigma_{x_1}^2(t) = \sigma_{x_1}^2(0) + \sigma_w^2 t$. That is, the variance of the system state itself diverges. As in the case of unmodeled states, the innovations vector $[z - H\hat{x}]$ will show effects of the "missing" dynamic disturbance noise.

Example 9.16 (Parametric Modeling Errors) Having the wrong parameters in the system dynamic coefficients F, state-transition matrix Φ, or output matrix H can and does bring about nonconvergence of the filter. This type of nonconvergence can be demonstrated by an example with the wrong period of a sinusoid in the output matrix:

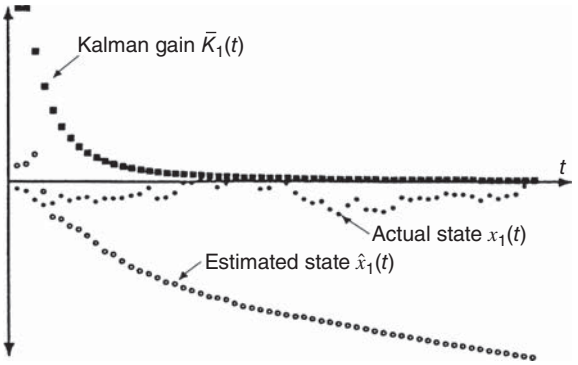

Figure 9.12 Diverging estimation error due to unmodeled process noise.

$$\begin{array}{cc}
\underline{\text{Real World}} & \underline{\text{Kalman Filter Model}} \\
\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = 0 & \begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = 0 \\
z = [\sin \Omega t | \cos \Omega t] \begin{bmatrix}x_1\\ x_2\end{bmatrix} + v & z = [\sin \omega t | \cos \omega t] \begin{bmatrix}x_1\\ x_2\end{bmatrix} + v \\
v = \text{white noise} & v = \text{white noise} \\
\text{No a priori information exists} & \text{No a priori information exists}
\end{array} \tag{9.59}$$

In this case, the optimum filter is the "least-squares" estimator acting over the measurement interval $(0, T)$. Since this is a continuous case,

$$J = \int_0^{\mathrm{T}} (z - H\hat{x})^{\mathrm{T}}(z - H\hat{x}) \quad dt \tag{9.60}$$

is the performance index being minimized. Its gradient

$$\frac{\partial J}{\partial \hat{x}} = 0 \quad \Rightarrow \quad 0 = 2\int_0^{\mathrm{T}} H^{\mathrm{T}} z \quad dt - 2\int_0^{\mathrm{T}} (H^{\mathrm{T}}H)\hat{x} \quad dt, \tag{9.61}$$

where $\hat{x}$ is unknown but constant. Therefore,

$$\hat{x} = \left[\int_0^{\mathrm{T}} H^{\mathrm{T}}H \quad dt\right]^{-1} \left[\int_0^{\mathrm{T}} H^{\mathrm{T}} z \quad dt\right], \tag{9.62}$$

where

$$H = [\sin \omega t | \cos \omega t]; \qquad z = [\sin \Omega t | \cos \Omega t] \begin{bmatrix}x_1\\ x_2\end{bmatrix} + v \tag{9.63}$$

and

$$\omega = 2\pi f = \frac{2\pi}{p}, \tag{9.64}$$

where p is the period of the sinusoid. For simplicity, let us choose the sampling time as $T = Np$, an integer multiple of the period, so that

$$\hat{x} = \begin{bmatrix} \frac{Np}{2} & 0 \\ 0 & \frac{Np}{2} \end{bmatrix}^{-1} \begin{bmatrix} \int_0^{Np} \sin(\omega t) z(t) & dt \\ \int_0^{Np} \cos(\omega t) z(t) & dt \end{bmatrix} = \begin{bmatrix} \frac{2}{Np} \int_0^{Np} z(t) \sin \omega t & dt \\ \frac{2}{Np} \int_0^{Np} z(t) \cos \omega t & dt \end{bmatrix}. \quad (9.65)$$

Concentrating on the first component $\hat{x}_1$, one can obtain its solution as

$$\begin{aligned} \hat{x}_1 = & \left\{ \frac{2}{Np} \left[\frac{\sin(\omega - \Omega)t}{2(\omega - \Omega)} - \frac{\sin(\omega + \Omega)t}{2(\omega + \Omega)} \right] \Big|_{t=0}^{t=Np} \right\} x_1 \\ & + \left\{ \frac{2}{Np} \left[\frac{-\cos(\omega - \Omega)t}{2(\omega - \Omega)} - \frac{\cos(\omega + \Omega)t}{2(\omega + \Omega)} \right] \Big|_{t=0}^{t=Np} \right\} x_2 \\ & + \frac{2}{Np} \int_0^{Np} v(t) \sin \omega t \quad dt. \end{aligned}$$

By setting $v = 0$ (ignoring the estimation error due to measurement noise), one obtains the result

$$\begin{aligned} \hat{x}_1 = & \frac{2}{Np} \left[\frac{\sin(\omega - \Omega)Np}{2(\omega - \Omega)} - \frac{\sin(\omega + \Omega)Np}{2(\omega + \Omega)} \right] x_1 \\ & + \frac{2}{Np} \left[\frac{1 - \cos(\omega - \Omega)Np}{2(\omega - \Omega)} + \frac{1 - \cos(\omega + \Omega)t}{2(\omega + \Omega)} \right] x_2. \end{aligned}$$

For the case that $\omega \to \Omega$,

$$\left.\begin{aligned} \frac{\sin(\omega - \Omega)Np}{2(\omega - \Omega)} &= \frac{Np}{2} \frac{\sin x}{x}\Big|_{x \to 0} = \frac{Np}{2}, \\ \frac{\sin(\omega + \Omega)Np}{2(\omega + \Omega)} &= \frac{\sin[(4\pi/p)Np]}{2\ \Omega} = 0, \\ \frac{1 - \cos(\omega - \Omega)Np}{2(\omega - \Omega)} &= \frac{1 - \cos x}{x}\Big|_{x \to 0} = 0, \\ \frac{1 - \cos(\omega + \Omega)Np}{2(\omega + \Omega)} &= \frac{1 - \cos[(4\pi/p)Np]}{2\ \Omega} = 0, \end{aligned}\right\} \quad (9.66)$$

and $\hat{x}_1 = x_1$. In any other case, $\hat{x}_1$ would be a biased estimate of the form

$$\hat{x}_1 = \Upsilon_1 x_1 + \Upsilon_2 x_2, \qquad \text{where} \quad \Upsilon_1 \neq 1, \quad \Upsilon_2 \neq 0. \quad (9.67)$$

Similar behavior occurs with $\hat{x}_2$.

Wrong parameters in the system and/or output matrices may not cause the filter covariance matrix or state vector to look unusual. However, the innovations vector $[z - H\hat{x}]$ will generally show detectable effects of nonconvergence.

This can only be cured by making sure that the right parameter values are used in the filter model. In the real world, this is often impossible to do precisely, since the "right" values are not known and can only be estimated. If the amount of degradation obtained is unacceptable, consider letting the questionable parameters become state variables to be estimated by extended (linearized nonlinear) filtering.

Example 9.17 (Parameter Estimation) This reformulation provides an example of a nonlinear estimation problem:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \end{bmatrix} = 0, \qquad x_3 = \Omega. \tag{9.68}$$

Here, something is known about Ω, but it is not known precisely enough. One must choose

$$\hat{x}_3(0) = \text{"best guess" value of } \Omega,$$
$$P_{33}(0) = \sigma^2_{\tilde{x}_3}(0) = \text{a measure of uncertainty in } \tilde{x}_3(0).$$

Nonlinearities in the real-world system also cause nonconvergence or even divergence of the Kalman estimates.

9.2.6 Analysis and Repair of Covariance Matrices

Covariance matrices must be *nonnegative definite*. By definition, their characteristic values must be nonnegative. However, if any of them are *theoretically* zero—or even close to zero—then there is always the risk that roundoff will cause some roots to become negative. If roundoff errors should produce an *indefinite* covariance matrix (i.e., one with both positive and negative characteristic values), then there is a way to replace it with a "nearby" nonnegative-definite matrix.

9.2.6.1 Testing for Positive Definiteness Checks that can be made for the definiteness of a symmetric matrix P include the following:

- If a *diagonal element* $a_{ii} < 0$, then the matrix is *not* positive definite, *but the matrix can have all positive diagonal elements and still fail to be positive definite*.
- If *Cholesky decomposition* $P = CC^{\mathrm{T}}$ fails due to a negative argument in a square root, the matrix is indefinite, or at least close enough to being indefinite that roundoff errors cause the test to fail.

- If *modified Cholesky decomposition* $P = UDU^{\mathrm{T}}$ produces an element $d_{ii} \leq 0$ in the diagonal factor D, then the matrix is not positive definite.
- *Singular value decomposition* yields all the characteristic values and vectors of a symmetric matrix. It is implemented in the MATLAB function `svd`.

The first of these tests is not very reliable unless the dimension of the matrix is 1.

Example 9.18 The following two 3×3 matrices have positive diagonal values and consistent correlation coefficients, yet neither of them is *positive definite* and the first is actually indefinite:

Matrix	Correlation Matrix	Singular Values
$\begin{bmatrix} 343.341 & 248.836 & 320.379 \\ 248.836 & 336.83 & 370.217 \\ 320.379 & 370.217 & 418.829 \end{bmatrix}$	$\begin{bmatrix} 1. & 0.73172 & 0.844857 \\ 0.73172 & 1. & 0.985672 \\ 0.844857 & 0.985672 & 1. \end{bmatrix}$	$\{1000,\ 100,\ -1\}$
$\begin{bmatrix} 343.388 & 248.976 & 320.22 \\ 248.976 & 337.245 & 369.744 \\ 320.22 & 369.744 & 419.367 \end{bmatrix}$	$\begin{bmatrix} 1. & 0.731631 & 0.843837 \\ 0.731631 & 1. & 0.983178 \\ 0.843837 & 0.983178 & 1. \end{bmatrix}$	$\{1000,\ 100,\ 0\}$

Repair of Indefinite Covariance Matrices The symmetric eigenvalue–eigenvector decomposition is the more informative of the test methods, because it yields the actual eigenvalues (characteristic values) and their associated eigenvectors (characteristic vectors). The characteristic vectors tell the combinations of states with equivalent negative variance. This information allows one to compose a matrix with the identical characteristic vectors but with the offending characteristic values "floored" at zero:

$$P = TDT^{\mathrm{T}} \text{ (symmetric eigenvalue–eigenvector decomposition)}, \tag{9.69}$$

$$D = \mathrm{diag}_i\{d_i\}, \tag{9.70}$$

$$d_1 \geq d_2 \geq d_3 \geq \cdots \geq d_n. \tag{9.71}$$

If

$$d_n < 0, \tag{9.72}$$

then replace P with

$$P^* = TD^*T^{\mathrm{T}}, \tag{9.73}$$

$$D^* = \mathrm{diag}_i\{d_i^*\}, \tag{9.74}$$

$$d_i^* = \begin{cases} d_i & \text{if} \quad d_i \geq 0, \\ 0 & \text{if} \quad d_i < 0. \end{cases} \tag{9.75}$$

9.3 PREFILTERING AND DATA REJECTION METHODS

9.3.1 Prefiltering

Prefilters perform data compression of the inputs to Kalman filters. They can be linear continuous, linear discrete, or nonlinear. They are beneficial for several purposes:

1. They allow a discrete Kalman filter to be used on a continuous system without "throwing away" information. For example, the integrate-and-hold prefilter shown in Figure 9.13 integrates over a period T, where T is a sampling time chosen sufficiently small that the dynamic states cannot change significantly between estimates.
2. They attenuate some states to the point that they can be safely ignored (in a suboptimal filter).
3. They can reduce the required iteration rate in a discrete filter, thereby saving computer time [2].
4. They tend to reduce the range of the dynamic variables due to added noise, so that the Kalman filter estimate is less degraded by nonlinearities.

9.3.1.1 Continuous (Analog) Linear Filters

Example 9.19 Continuous linear filters are usually used for the first three purposes and must be inserted before the sampling process. An example of the continuous linear prefilter is shown in Figure 9.14. An example of such a prefilter is a digital voltmeter (DVM). A DVM is essentially a time-gated averager with sampler and quantizer.

Thus the input signal is continuous and the output discrete. A functional representation is given in Figures 9.15–9.17, where

$$\Delta T = \text{sampling interval},$$

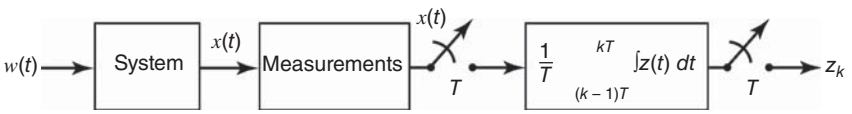

Figure 9.13 Integrate-and-hold prefilter.

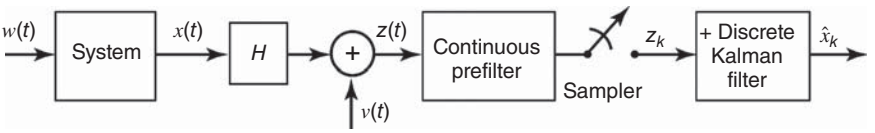

Figure 9.14 Continuous linear prefiltering and sampling.

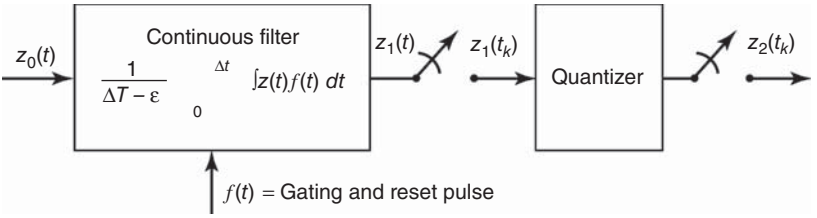

Figure 9.15 Block diagram of DVM.

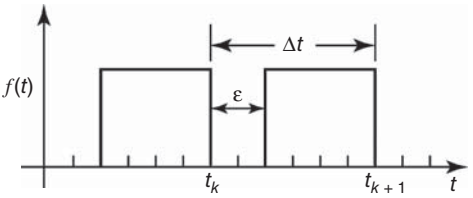

Figure 9.16 DVM gating waveform.

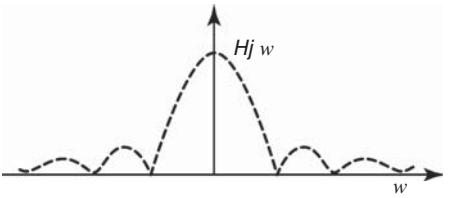

Figure 9.17 DVM frequency response.

ε = dead time for rezeroing the integrator,

$\Delta T - \varepsilon$ = averaging time,

and the output

$$z^1(t_i) = \frac{1}{\Delta T - \varepsilon} \int_{t_i - \Delta T + \varepsilon}^{t_i} z(t) \quad dt. \tag{9.76}$$

It can be shown that the frequency response of the DVM is

$$|H(j\omega| = \left| \frac{\sin \omega[(\Delta T - \varepsilon)/2]}{\omega[(\Delta T - \varepsilon)/2]} \right| \quad \text{and} \quad \theta(j\omega) = -\omega \left(\frac{\Delta T - \varepsilon}{2} \right). \tag{9.77}$$

With white-noise continuous input, the output is a white-noise sequence (since the averaging intervals are nonoverlapping). With an exponentially correlated random

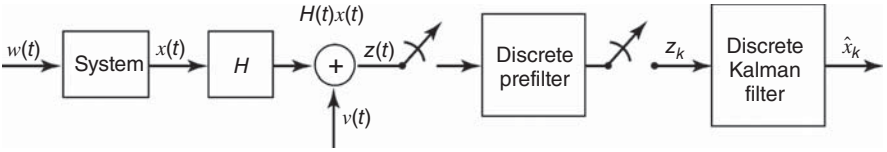

Figure 9.18 Discrete linear prefilter.

continuous input with correlation time τ_c and autocovariance

$$\psi_z(\tau) = \sigma_z^2 e^{-(|\tau|/\tau_c)}, \tag{9.78}$$

the variance and autocorrelation function of the output random sequence can be shown to be

$$\left.\begin{aligned} \sigma_{z^1}^2 &= \psi_{z^1 z^1}(0) = f(u)\sigma_z^2, \\ \psi(j-i) &= g(u)\sigma_z^2 e^{-(j-i)\frac{\Delta T}{\tau_c}}, \end{aligned}\right\} \quad \begin{aligned} u &= \tfrac{\Delta T - \varepsilon}{T_c}, \\ f(u) &= \tfrac{2}{u^2}(e^{-u} + u - 1) \\ g(u) &= \tfrac{e^u + e^{-u} - 2}{u^2}. \end{aligned} \tag{9.79}$$

9.3.1.2 Discrete Linear Filters These can be used effectively for the attenuation of the effects of some of the state variables to the point that they can be safely ignored. (However, the sampling rate input to the filter must be sufficiently high to avoid aliasing.)

Note that discrete filters can be used for the third purpose to reduce the discrete filter iteration rate. The input sampling period can be chosen shorter than the output sampling period. This can greatly reduce the computer (time) load (see Figure 9.18).

Example 9.20 For the simple digital averager shown in Figure 9.19, let

$$z^1(t_i^1) \equiv \frac{1}{N} \sum_{j=i-N+1}^{i} z(t_j), \quad t_i^1 = iT^1, \tag{9.80}$$

which is the average of N adjacent samples of $z(t_j)$. Note that $z^1(t_i^1)$ and $z^1(t_{i+1}^1)$ use nonoverlapping samples of $z(t_j)$. Then it can be shown that the frequency response is

$$|H(j\omega)| = \frac{|\sin(N\omega T/2)|}{N|\sin(\omega T/2)|}, \theta(j\omega) = -\left(\frac{N-1}{2}\right)\omega T. \tag{9.81}$$

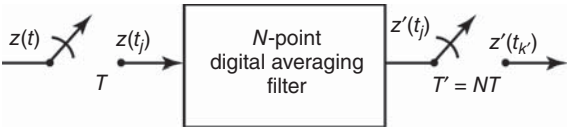

Figure 9.19 Discrete linear averager.

9.3.2 Data Rejection

If adequate knowledge of the innovations vector $[z - H\hat{x}]$ exists, nonlinear "data rejection filters" can be implemented. Statistical approaches were discussed in Section 9.2.1. Some simple examples are cited below.

Example 9.21 (Data Rejection Filters) For excess amplitude,

$$\text{If } |(z - H\hat{x})_i| > A_{\max}, \text{ then reject data.} \tag{9.82}$$

For excess rate (or change),

$$\text{If } |(z - H\hat{x})_{i+1} - (z - H\hat{x})_i| > \delta A_{\max}, \text{ then reject data.} \tag{9.83}$$

Many other ingenious techniques have been used, but they usually depend on the specifics of the problem.

9.4 STABILITY OF KALMAN FILTERS

The dynamic stability of a system usually refers to the behavior of the state variables, not the estimation errors. This applies as well to the behavior of the homogeneous part of the filter equations. However, the mean-squared estimation errors may remain bounded, even if the system is unstable.[4]

If the actual measurement processing in the Kalman filter state equations is neglected, then the resulting equations characterize the stability of the filter itself. In the continuous case, these equations are

$$\dot{\hat{x}}(t) = [F(t) - \overline{K}(t)H(t)]\hat{x}(t), \tag{9.84}$$

and in the discrete case,

$$\begin{aligned}\hat{x}_{k(+)} &= \Phi_{k-1}\hat{x}_{k-1(+)} - \overline{K}_k H_k \Phi_{k-1}\hat{x}_{k-1(+)} \\ &= [I - \overline{K}_k H_k]\Phi_{k-1}\hat{x}_{k-1(+)}.\end{aligned} \tag{9.85}$$

[4]See, for example, Gelb et al. [3, pp. 22, 31, 36, 53, 72] or Maybeck [4, p. 278].

The solution of the filter Equation 9.84 or 9.85 is uniformly asymptotically stable, which means bounded input-bounded output (BIBO) stability. Mathematically, it implies that

$$\lim_{t\to\infty} \|\hat{x}(t)\| = 0 \tag{9.86}$$

or

$$\lim_{k\to\infty} \|\hat{x}_k(+)\| = 0, \tag{9.87}$$

no matter what the initial conditions are. In other words, the filter is uniformly asymptotically stable if the system model is stochastically controllable and observable. See Chapter 5 for the solution of the matrix Riccati equation $P(t)$ or $P_k(+)$ uniformly bounded from above for large t or $\overline{K}$ independent of $P(0)$. Bounded Q, R (above and below), and bounded F (or Φ) will guarantee stochastic controllability and observability.

The most important issues relating to stability are described in the sections on unmodeled effects, finite wordlength, and other errors (Section 9.2).

9.5 SUBOPTIMAL AND REDUCED-ORDER FILTERS

9.5.1 Suboptimal Filters

The Kalman filter has a reputation for being robust against certain types of modeling errors, such as those in the assumed values of the statistical parameters R and Q. This reputation is sometimes tested by deliberate simplification of the known (or, at least, "believed") system model. The motive for these actions is usually to reduce implementation complexity by sacrificing some optimality. The result is called a *suboptimal filter*.

9.5.1.1 Rationale for Suboptimal Filtering It is often the case that real hardware is nonlinear but, in the filter model, approximated by a linear system. The algorithms developed in Chapters 5–7 will provide suboptimal estimates. These are

1. Kalman filters (linear optimal estimate),
2. linearized Kalman filters, and
3. extended Kalman filters.

Even if there is good reason to believe that the real hardware is truly linear, there may still be reasons to consider suboptimal filters. Where there is doubt about the absolute certainty of the model, there is always a motivation to meddle with it, especially if meddling can decrease the implementation requirements. Optimal filters are generally demanding on computer throughput, and optimality is unachievable if the required computer capacity is not available. Suboptimal filtering can reduce the requirements for computer memory, throughput, and cost. A suboptimal filter design may be "best" if factors other than theoretical filter performance are considered in the trade-offs.

9.5.1.2 Suboptimal Filtering Techniques These techniques can be divided into three categories:

1. modifying the optimal gain $\overline{K}_k$ or $\overline{K}(t)$,
2. modifying the filter model, and
3. other techniques.

9.5.1.3 Evaluating Suboptimal Filters The covariance matrix P may not represent the actual estimation error and the estimates may be biased. In the following section, the dual-state technique for evaluating performance of linear suboptimal filters will be discussed.

Modification of $\overline{\boldsymbol{K}}(t)$ *or* $\overline{\boldsymbol{K}}_{\boldsymbol{k}}$ Consider the system

$$\begin{aligned} \dot{x} &= Fx + Gw, \qquad Q = \text{cov}(w), \\ z &= Hx + v, \quad R = \text{cov}(v), \end{aligned} \tag{9.88}$$

with state-transition matrix Φ.

The state estimation portion of the optimal linear estimation algorithm is then, for the continuous case,

$$\dot{\hat{x}} = F\hat{x} + \overline{K}(t)[z - H\hat{x}] \text{ with } \hat{x}(0) = \hat{x}_0, \tag{9.89}$$

and for the discrete case,

$$\hat{x}_{k(+)} = \Phi_{k-1}\hat{x}_{k-1(+)} + \overline{K}_k[z_k - H_k(\Phi_{k-1}\hat{x}_{k-1(+)})] \tag{9.90}$$

with initial conditions $\hat{x}_0$.

Schemes for retaining the structure of these algorithms using modified gains include the Wiener filter and approximating functions.

First, consider the *Wiener filter*, which is a useful suboptimal filtering method when the gain vector $\overline{K}(t)$ is time varying but quickly reaches a constant nonzero steady-state value. Typical settling behaviors are shown in Figure 9.20.

The Wiener filter results from the approximation

$$\overline{K}(t) \approx \overline{K}(\infty). \tag{9.91}$$

If, in addition, the matrices F and H are time invariant, the matrix of transfer functions characterizing the Wiener filter can easily be computed:

$$\dot{\hat{x}} = F\hat{x} + \overline{K}(\infty)[z - H\hat{x}] \tag{9.92}$$

$$\Rightarrow \frac{\hat{x}(s)}{z(s)} = [sI - F + \overline{K}(\infty)H]^{-1}. \tag{9.93}$$

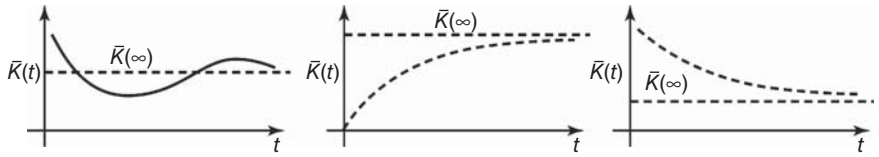

Figure 9.20 Settling of Kalman gains.

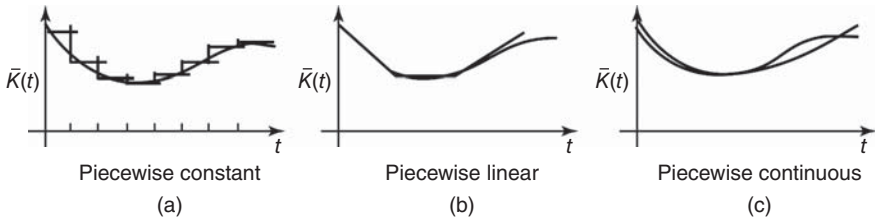

Figure 9.21 Approximating time-varying Kalman gains. (a) Piecewise constant, (b) Piecewise linear, and (c) Piecewise continuous.

The corresponding steady-state sinusoidal frequency response matrix is

$$\frac{|\hat{x}(j\omega)|}{|z(j\omega)|} = |[(j\omega)I - F + \overline{K}(\infty)H]^{-1}|. \tag{9.94}$$

Among the advantages of the Wiener filter are that—its structure being identical to conventional filters—all the tools of pole-zero, frequency response, and transient response analysis using Laplace transforms can be employed to gain "engineering insight" into the behavior of the filter. Among the disadvantages of this approach is that it cannot be used if $\overline{K}(\infty) \neq$ constant or $\overline{K}(\infty) = 0$. The typical penalty is poorer transient response (slower convergence) than with optimal gains.

The second scheme for retaining the structure of the algorithms using modified gains is that of *approximating functions*. The optimal time-varying gains $\overline{K}_{\text{OP}}(t)$ are often approximated by simple functions.

For example, one can use piecewise constant approximation, $\overline{K}_{\text{pwc}}$, a piecewise linear approximation, $\overline{K}_{\text{pwl}}$, or a curve fit using smooth functions $\overline{K}_{\text{CF}}$, as shown in Figure 9.21:

$$\overline{K}_{\text{CF}}(t) = C_1 e^{-a_1 t} + C_2(1 - e^{-a_2 t}). \tag{9.95}$$

The advantages of approximating functions over the Wiener filter are that this can handle cases where $\overline{K}(\infty)$ is not constant or $\overline{K}(\infty) = 0$. A result closer to optimal performance is obtained.

9.5.1.4 Modifying Filter Models Let the real-world model (actual system S) be linear,

$$\dot{x}^s = F_s x_s + G_s w^s, \qquad z^s = H_s x_s + v^s.$$

The filter model of the system will, in general, be (intentionally or unintentionally) different:

$$\dot{x}_F = F_F x_F + G_F w_F, \qquad z_F = H_F x_F + v_F.$$

Usually, the intent is to make the filter model less complex than the actual system. This can be done by ignoring some states, prefiltering to attenuate some states, decoupling states, or with frequency-domain approximations. Ignoring some states reduces model complexity and provides a suboptimal design. Often, however, little performance degradation occurs.

Example 9.22 In this example, one combines two nonobservable states with identical propagation into z.

$$\begin{aligned} \text{From:} \quad & \dot{x}_1 = -ax_1, \\ & \dot{x}_2 = -ax_2, \\ & z = [23]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \end{aligned} \qquad \begin{aligned} \text{To:} \quad & \dot{x}^1 = -ax^1, \\ & z = x^1 + v, \end{aligned} \tag{9.96}$$

Of course, $x^1 \equiv 2x_1 + 3x_2$ and the a priori information must be represented as

$$\begin{aligned} \hat{x}^1(0) &= 2\hat{x}_1(0) + 3\hat{x}_2(0), \\ P_{x^1x^1}(0) &= 4P_{x_1x_1}(0) + 12P_{x_1x_2}(0) + 9P_{x_2x_2}(0). \end{aligned}$$

Example 9.23 Continuing where Example 9.13 left off, one can combine two states if they are "close functions" over the entire measurement interval:

$$\begin{aligned} \text{From:} \quad & \dot{x}_1 = 0, \\ & \dot{x}_2 = 0, \\ & z = [t|\sin t]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \end{aligned} \qquad \begin{aligned} \text{To:} \quad & \dot{x}^1 = 0, \\ & z = tx^1 + v, \end{aligned} \tag{9.97}$$

where the a priori information on x^1 must be formulated and where z is available on the interval $(0, \pi/20.)$

Example 9.24 (Ignoring Small Effects)

$$\begin{aligned} \text{From:} \quad & \dot{x}_1 = -ax_1, \\ & \dot{x}_2 = -bx_2, \\ & z = x_1 + x_2 + v \text{ on } (0, T), \end{aligned} \qquad \begin{aligned} \text{To:} \quad & \dot{x}_2 = -bx_2, \\ & z = x_2 + v, \end{aligned} \tag{9.98}$$

with

$$E(x_1^2) = 0.1, \quad E(x_2^2) = 10.0, \qquad \text{desired} \quad P_{22}(T) = 1.0. \tag{9.99}$$

Example 9.25 (Relying on Time Orthogonality of States)

$$
\begin{aligned}
\text{From:}\quad & \dot{x}_1 = 0, \\
& \dot{x}_2 = 0, \\
& z = [1 \mid \sin t] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + v, \\
& z \text{ available on } (0, T) \text{ with } T \text{ large} \\
& P_{22}(0) = \text{large}.
\end{aligned}
\qquad
\begin{aligned}
\text{To:}\quad & \dot{x}_2 = 0, \\
& z = (\sin t) x_2 + v,
\end{aligned}
\tag{9.100}
$$

9.5.1.5 Prefiltering to Simplify the Model A second technique for modifying the filter model of the real world is to ignore states after prefiltering to provide attenuation. This is illustrated in Figure 9.22.

Of course, prefiltering has deleterious side effects, too. It may require other changes to compensate for the measurement noise v becoming time correlated after passing through the prefilter and to account for some "distortion" of those states in the passband of the prefilter (e.g., amplitude and phase of sinusoidals and wave shape of signals). The filter may be modified as shown in Figure 9.23 to compensate for such effects. Hopefully, the net result is a smaller filter than before.

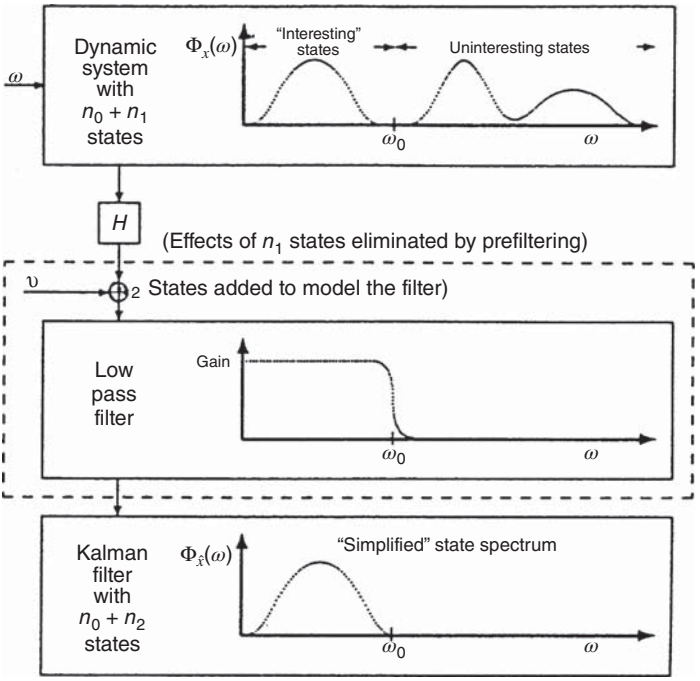

Figure 9.22 Low pass prefiltering for model simplification.

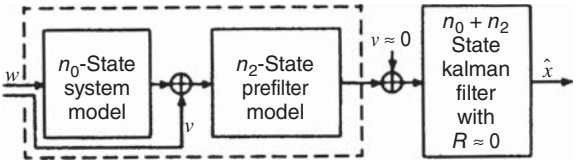

Figure 9.23 Modified system model for low pass prefiltering.

Example 9.26 Consider the second-order system with coupling coefficient τ, represented by the equations

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\alpha_1 & 0 \\ \tau & -\alpha_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix},$$
$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}.$$

If τ is sufficiently small, one can treat the system as two separate uncoupled systems with two separate corresponding Kalman filters:

$$\begin{aligned} \dot{x}_1 &= -\alpha_1 x_1 + w_1, \quad & \dot{x}_2 &= -\alpha_2 x_2 + w_2, \\ z_1 &= x_1 + v_1, & z_2 &= x_2 + v_2. \end{aligned} \tag{9.101}$$

The advantage of this decoupling method is a large reduction in computer load. However, the disadvantage is that the estimation error has increased variance and can be biased.

9.5.1.6 Frequency-domain Approximations These can be used to formulate a suboptimal filter for a system of the sort

$$\dot{x} = Fx + Gw, \qquad z = Hx + v.$$

Often, some of the states are stationary random processes whose power spectral densities can be approximated as shown in Figure 9.24.

A filter designed for the approximate spectrum may well use fewer states.

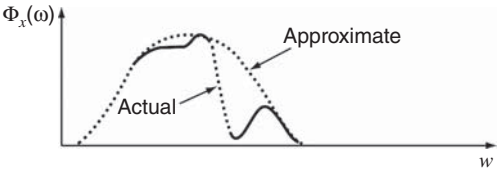

Figure 9.24 Frequency-domain approximation.

Example 9.27 Sometimes the general structure of the random process model is known, but the parameters are not known precisely:

$$\dot{x} = -\alpha x + w \text{ and } \sigma_w \text{ are "uncertain."} \tag{9.102}$$

Replacing this model by a random walk

$$\dot{x} = w \tag{9.103}$$

in conjunction with "sensitivity studies" will often allow a judicious choice for σ_w with small sensitivity to α uncertainties and small degradation in filter performance.

Example 9.28 (White-Noise Approximation of Broadband Noise) If the system-driving noise and measurement noise are "broadband" but with sufficiently flat PSDs, they can be replaced by "white-noise" approximations, as illustrated in Figure 9.25.

Least-Squares Filters Among the other techniques used in practice to intentionally suboptimize linear filters is least-squares estimation. It is equivalent to Kalman filtering if there are no state dynamics ($F = 0$ and $Q = 0$) and is often considered as a candidate for a suboptimal filter if the influence of the actual values of Q and F (or Φ) on the values of the Kalman gain is small.

Observer Methods These simpler filters can be designed by choosing the eigenvalues of a filter of special structure. The design of suboptimal filters using engineering insight is often possible. Ingenuity based on physical insights with regard to design of suboptimal filters is considered an art, not a science, by some practitioners.

9.5.2 Dual-State Evaluation of Suboptimal Filters

9.5.2.1 Dual-State Analysis This form of analysis takes its name from the existence of two views of reality:

1. The so-called *system model* (or "truth model") of the actual system under study. This model is used to generate the observations input to the suboptimal filter. It should be a reasonably complete model of the actual system under consideration, including all known phenomena that are likely to influence the performance of the estimator.

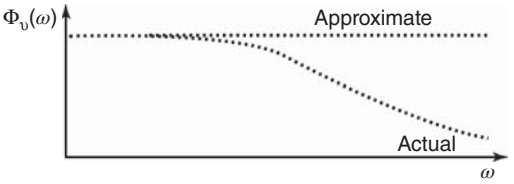

Figure 9.25 White-noise approximation of broadband noise.

2. The *filter model*, which is a reduced-order version of the system model, it is usually constrained to contain all the states in the *domain* of the measurement sensitivity matrix (i.e., the states that contribute to the measurement) and possibly other state components of the system model as well. The filter model is to be used by a proposed filter implementation with significantly reduced computational complexity, but (hopefully) not greatly reduced fidelity. The performance of the reduced-order filter implementation is usually measured by how well its estimates agree with those of the actual system state. It may not estimate *all* of the components of the state vector of the system model, however. In that case, the evaluation of its estimation accuracy is restricted to just the common state vector components.

Performance analysis of suboptimal filter errors requires the simultaneous consideration of both (system and filter) state vectors, which are combined into one *dual-state vector*.

9.5.2.2 Notation Let us use a superscript notation to distinguish between these models. A superscript S will denote the *system model*, and a superscript F will denote the filter model, as illustrated later in Figure 9.31.[5] There are two commonly used definitions for the dual-state vector:

1. $x_k^{\text{dual}} = \begin{bmatrix} x_k^S \\ \hat{x}_{k(+)}^F \end{bmatrix}$ (concatenation of the two vectors) and
2. $\tilde{x}_k^{\text{dual}} = \begin{bmatrix} \tilde{x}_{k(+)}^S \\ x_k^S \end{bmatrix}, \tilde{x}_{k(+)}^S = \hat{x}_{k(+)}^F - x_k^S.$

In the second definition, the filter state vector x^F may have lower dimension than the system state vector x^S—due to neglected state components of the system in the suboptimal filter model. In that case, the missing components in x^F can be padded with zeros to make the dimensions match.

In dual-state analysis for evaluating suboptimal filters, let the following definitions apply:

Actual System	Filter Model	
$\dot{x}^S = F_S x^S + G_S w^S$	$\dot{x}^F = F_F x^F + G_F w^F$	(9.104)
$z^S = H_S x^S + v^S$	$z^F = H_F x^F + v^F$	
$z^F = z^S,$	$H_F = H_S, v^F = v^S,$	(9.105)

[5]The "system" model is called the *truth* model in the figure. That nomenclature here would lead us to use a superscript T, however, and that notation is already in use to denote transposition.

$$\left.\begin{array}{l}\eta_k^S \equiv \int_{t_{k-1}}^{t_k} \Phi_S(t_k - \tau) G_S(\tau) w^S(\tau)\, \mathrm{d}\tau, \\ \hat{x}_{k(+)}^F = \Phi_F \hat{x}_{k-1(+)}^F + \overline{K}_k [z_k^F - H_F \Phi_F \hat{x}_{k-1(+)}^F], \\ \Phi_S \equiv \text{System state-transition matrix}, \\ \Phi_F \equiv \text{State-transition matrix of filter model}, \\ Q_S \equiv \text{cov}(w^S), \\ Q_F \equiv \text{cov}(w^F), \end{array}\right\} \tag{9.106}$$

Let $\Gamma_k \equiv (I - \overline{K}_k H_F)$, and let

$$A \equiv \begin{bmatrix} \Phi_F & \Phi_F - \Phi_S \\ 0 & \Phi_S \end{bmatrix}, \tag{9.107}$$

$$B \equiv \begin{bmatrix} \Gamma_k & 0 \\ 0 & I \end{bmatrix}. \tag{9.108}$$

Let the "prediction" estimation error be

$$\tilde{x}_k^S(+) \equiv \hat{x}_k^F(+) - x_k^S \tag{9.109}$$

and let the "filtered" estimation error be

$$\tilde{x}_{k-1}^S(+) \equiv \hat{x}_{k-1}^F(+) - x_{k-1}^S. \tag{9.110}$$

Then the prediction error equation is

$$\begin{bmatrix} \tilde{x}_{k(-)}^S \\ x_k^S \end{bmatrix} = A \begin{bmatrix} \tilde{x}_{k-1(+)}^S \\ x_{k-1}^S \end{bmatrix} + \begin{bmatrix} -\eta_{k-1}^S \\ \eta_{k-1}^S \end{bmatrix} \tag{9.111}$$

and the filtered error equation is

$$\begin{bmatrix} \tilde{x}_{k-1(+)}^S \\ x_{k-1}^S \end{bmatrix} = B \begin{bmatrix} \tilde{x}_{k-1(-)}^S \\ x_{k-1}^S \end{bmatrix} + \begin{bmatrix} \overline{K}_{k-1} v_{k-1}^S \\ 0 \end{bmatrix}. \tag{9.112}$$

Taking the expected value E and the covariance of Equations 9.111 and 9.112 yields the following recursive relationships:

$$\left.\begin{array}{l} E_{\text{cov}} = A \cdot P_{\text{cov}} \cdot A^T + \text{cov}L \\ P_{\text{cov}} = B \cdot E_{\text{cov}} \cdot B^T + \text{cov}P \end{array}\right\} \text{(covariance propagation)}, \tag{9.113}$$

$$\left.\begin{array}{l} E_{\text{DX}} = A \cdot P_{\text{DX}} \\ P_{\text{DX}} = B \cdot E_{\text{DX}} \end{array}\right\} \text{(bias propagation)}, \tag{9.114}$$

where the newly introduced symbols are defined as follows:

$$
\begin{aligned}
\mathrm{E}_{\mathrm{cov}} &\equiv \mathrm{cov}\begin{bmatrix}\tilde{x}^S_{k(-)}\\ x^S_k\end{bmatrix} \text{ (predicted dual-state vector)},\\
\mathrm{cov}L &\equiv \mathrm{cov}\begin{bmatrix}-\eta^S_{k-1}\\ \eta^S_{k-1}\end{bmatrix},\\
P_{\mathrm{cov}} &\equiv \mathrm{cov}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} \text{ (filtered covariance)},\\
\mathrm{cov}P &\equiv \mathrm{cov}\begin{bmatrix}\overline{K}_{k-1}v^S_{k-1}\\ 0\end{bmatrix},\\
&= \begin{bmatrix}\overline{K}_{k-1}\mathrm{cov}(v^S_{k-1})\overline{K}^{\mathrm{T}}_{k-1} & 0\\ 0 & 0\end{bmatrix},\\
P_{\mathrm{DX}} &= \mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} \text{ (expected value of filtered dual-state vector)},\\
\mathrm{E}_{\mathrm{DX}} &= \mathrm{E}\begin{bmatrix}\tilde{x}^S_{k(-)}\\ x^S_k\end{bmatrix} \text{ (expected value of predicted dual-state vector)}.
\end{aligned}
\tag{9.115}
$$

The suboptimal estimate $\hat{x}$ can be biased. The estimation error covariance can depend on the system state covariance. To show this, one can use the dual-state equations. The bias propagation equations

$$E_{\mathrm{DX}} = A \cdot P_{\mathrm{DX}}, P_{\mathrm{DX}} = B \cdot E_{\mathrm{DX}}$$

become

$$\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k(-)}\\ x^S_k\end{bmatrix} = A\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} \tag{9.116}$$

and

$$\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(+)}\\ x^S_{k-1}\end{bmatrix} = B\mathrm{E}\begin{bmatrix}\tilde{x}^S_{k-1(-)}\\ x^S_{k-1}\end{bmatrix}. \tag{9.117}$$

Clearly, if $\mathrm{E}[\tilde{x}^S_k(+)] \neq 0$, then the estimate becomes biased. If

$$\Phi_F \neq \Phi_S \tag{9.118}$$

and

$$\mathrm{E}(x^S) \neq 0, \tag{9.119}$$

which often is the case (e.g., x^S may be a deterministic variable, so that $E(x^S) = x^S \neq 0, x^S$ may be a random variable with nonzero mean). Similarly, an examination of the covariance propagation equations

$$E_{\text{cov}} = A(P_{\text{cov}})A^{\text{T}} + \text{cov}L$$
$$P_{\text{cov}} = B(E_{\text{cov}})B^{\text{T}} + \text{cov}P$$

show that $\text{cov}[\tilde{x}_k^S(-)]$ depends on $\text{cov}(x_k^S)$. (See Problem 7.5.)

If

$$\Phi_F \neq \Phi_S \tag{9.120}$$

and

$$\text{cov}(x^S) \neq 0, \tag{9.121}$$

which often is the case with the suboptimal filter, the estimate $\hat{x}$ is unbiased and the estimation error covariance is independent of the system state.

9.6 SCHMIDT–KALMAN FILTERING

9.6.1 Historical Background

Stanley F. Schmidt was an early and successful advocate of Kalman filtering. He was working at the NASA Ames Laboratory in Mountain View, California, when Kalman presented his results there in 1959. Schmidt immediately began applying it to a problem then under study at Ames, which was the space navigation problem (i.e., trajectory estimation) for the upcoming Apollo project for manned exploration of the moon. (In the process, Schmidt discovered what is now called *extended Kalman filtering*.) Schmidt was so impressed with his results that he set about proselytizing his professional colleagues and peers to try Kalman filtering.

Schmidt also derived and evaluated many practical methods for improving the numerical stability of the procedures used and for reducing the computational requirements of Kalman filtering. Many of these results were published in journal articles, technical reports, and books. In Reference 5, Schmidt presents an approach (now called *Schmidt–Kalman filtering*) for reducing the computational complexity of Kalman filters by eliminating some of the computation for "nuisance variables," which are state variables that are of no interest for the problem at hand—except that they are part of the system state vector.

Schmidt's approach is suboptimal, in that it sacrifices estimation performance for computational performance. It enabled Kalman filters to be approximated so that they could be implemented in real time on the computers of that era (the mid-1960s). However, it still finds useful application today for implementing Kalman filters on small embedded microprocessors.

The types of nuisance variables that find their way into the Kalman filter state vector include those used for modeling correlated measurement noise (e.g., colored, pastel, or random-walk noise). We generally have no interest in the memory state of such noise. We just want to filter it out.

Because the dynamics of measurement noise are generally *not* linked to the other system state variables, these added state variables are not dynamically coupled to the other state variables. That is, the elements in the dynamic coefficient matrix linking

the two state variable types (states related to correlated measurement noise and states not related to correlated measurement noise) are zero. In other words, if the ith state variable is of one type and the jth state variable is of the other type, then the element f_{ij} in the ith row and jth column of the dynamic coefficient matrix F will always be zero.

Schmidt was able to take advantage of this, because it means that the state variables could be reordered in the state vector such that the nuisance variables appear last. The resulting dynamic equation then has the form

$$\frac{d}{dt}\mathbf{x}(t) = \begin{bmatrix} F_{\varepsilon}(t) & 0 \\ 0 & F_{\nu}(t) \end{bmatrix} \mathbf{x}(t) + \mathbf{w}(t) \tag{9.122}$$

such that F_{ν} represents the dynamics of the nuisance variables and F_{ε} represents the dynamics of the other state variables.

It is this partitioning of the state vector that leads to the reduced-order, suboptimal filter called the *Schmidt–Kalman filter*.

9.6.2 Derivation

9.6.2.1 Partitioning the Model Let[6]

$n = n_{\varepsilon} + n_{\nu}$ be the total number of state variables,

n_{ε} be the number of essential variables, whose values are of interest for the application, and

n_{ν} be the number of *nuisance* variables, whose values are of no intrinsic interest and whose dynamics are not coupled with those of the essential state variables.

Then the state variables can be reordered in the state vector such that the essential variables precede the nuisance variables:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n_{\varepsilon}} \\ x_{n_{\varepsilon+1}} \\ x_{n_{\varepsilon+2}} \\ x_{n_{\varepsilon+3}} \\ \vdots \\ x_{n_{\varepsilon}+n_{\nu}} \end{bmatrix} \begin{matrix} \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} \text{essential variables} \\ \left.\begin{matrix} \\ \\ \\ \\ \\ \end{matrix}\right\} \text{nuisance variables} \end{matrix} \tag{9.123}$$

[6]This derivation follows that in Reference 6.

$$= \left[\frac{\mathbf{x}_\varepsilon}{\mathbf{x}_\nu}\right], \tag{9.124}$$

where the state vector has been partitioned into a subvector $\mathbf{x}_\varepsilon$ of the essential state variables and a subvector $\mathbf{x}_\nu$ of nuisance variables.

9.6.2.2 Partitioning Dynamic Models We know that these two state variable types are not linked dynamically, so that the system dynamic model has the form

$$\frac{d}{dt}\left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_\nu(t)}\right] = \left[\frac{F_\varepsilon(t) \mid 0}{0 \mid F_\nu(t)}\right]\left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_\nu(t)}\right] + \left[\frac{\mathrm{w}_\varepsilon(t)}{\mathrm{w}_\nu(t)}\right] \tag{9.125}$$

in continuous time, where the process noise vectors $\mathbf{w}_\varepsilon$ and $\mathbf{w}_\nu$ are uncorrelated. That is, the covariance matrix of process noise

$$Q = \left[\frac{Q_{\varepsilon\varepsilon} \mid 0}{0 \mid Q_{\nu\nu}}\right] \tag{9.126}$$

for the continuous-time model (as well as for the discrete-time model). That is, the cross-covariance block $Q_{\varepsilon\nu} = 0$.

9.6.2.3 Partitioned Covariance Matrix The covariance matrix of estimation uncertainty (the dependent variable of the Riccati equation) can also be partitioned as

$$P = \left[\frac{P_{\varepsilon\varepsilon} \mid P_{\varepsilon\nu}}{P_{\nu\varepsilon} \mid P_{\nu\nu}}\right], \tag{9.127}$$

where

the block $P_{\varepsilon\varepsilon}$ is dimensioned $n_\varepsilon \times n_\varepsilon$,
the block $P_{\varepsilon\nu}$ is dimensioned $n_\varepsilon \times n_\nu$,
the block $P_{\nu\varepsilon}$ is dimensioned $n_\nu \times n_\varepsilon$, and
the block $P_{\nu\nu}$ is dimensioned $n_\nu \times n_\nu$.

9.6.2.4 Temporal Covariance Update The corresponding state-transition matrix for the discrete-time model will then be of the form

$$\Phi_\kappa = \left[\frac{\Phi_{\varepsilon\kappa} \mid 0}{0 \mid \Phi_{\nu\kappa}}\right], \tag{9.128}$$

$$\Phi_{\varepsilon k} = \exp\left(\int_{t_{k-1}}^{t_k} F_\varepsilon(t)\, dt\right), \tag{9.129}$$

$$\Phi_{\nu k} = \exp\left(\int_{t_{k-1}}^{t_k} F_\nu(t)\, dt\right), \tag{9.130}$$

and the temporal update of P will have the partitioned form

$$\left[\begin{array}{c|c} P_{\varepsilon\varepsilon k+1\,-} & P_{\varepsilon\nu k+1\,-} \\ \hline P_{\nu\varepsilon k+1\,-} & P_{\nu\nu k+1\,-} \end{array}\right] = \left[\begin{array}{c|c} \Phi_{\varepsilon k} & 0 \\ \hline 0 & \Phi_{\nu\,k} \end{array}\right] \left[\begin{array}{c|c} P_{\varepsilon\varepsilon k+} & P_{\varepsilon\nu k+} \\ \hline P_{\nu\varepsilon k+} & P_{\nu\nu k+} \end{array}\right] \left[\begin{array}{c|c} \Phi_{\varepsilon k}^{\mathrm{T}} & 0 \\ \hline 0 & \Phi_{\nu\,k}{}^{\mathrm{T}} \end{array}\right] + \left[\begin{array}{c|c} Q_{\varepsilon\varepsilon} & 0 \\ \hline 0 & Q_{\nu\nu} \end{array}\right], \tag{9.131}$$

or, in terms of the individual blocks,

$$P_{\epsilon\epsilon k+1-} = \Phi_{\epsilon k} P_{\epsilon\epsilon k+} \Phi_{\epsilon k}^{\mathrm{T}} + Q_{\epsilon\epsilon}, \tag{9.132}$$

$$P_{\epsilon\upsilon k+1-} = \Phi_{\epsilon k} P_{\epsilon\upsilon k+} \Phi_{\upsilon k}^{\mathrm{T}}, \tag{9.133}$$

$$P_{\upsilon\epsilon k+1-} = \Phi_{\upsilon k} P_{\upsilon\epsilon k+} \Phi_{\epsilon k}^{\mathrm{T}}, \tag{9.134}$$

$$P_{\upsilon\upsilon k+1-} = \Phi_{\upsilon k} P_{\upsilon\upsilon k+} \Phi_{\upsilon k}^{\mathrm{T}} + Q_{\upsilon\upsilon}. \tag{9.135}$$

9.6.2.5 Partitioned Measurement Sensitivity Matrix With this partitioning of the state vector, the measurement model will have the form

$$z = [H_\varepsilon \quad | \quad H_\upsilon] \left[\frac{\mathbf{x}_\varepsilon(t)}{\mathbf{x}_\upsilon(t)}\right] + \upsilon \tag{9.136}$$

$$= \underbrace{H_\epsilon \mathbf{x}_\varepsilon}_{\substack{\text{Essential}\\ \text{state}\\ \text{dependence}}} + \underbrace{H_\upsilon \mathbf{x}_\upsilon}_{\substack{\text{Correlated}\\ \text{noise}}} + \underbrace{\upsilon}_{\substack{\text{Uncorrelated}\\ \text{noise}}}. \tag{9.137}$$

9.6.3 Schmidt–Kalman Gain

9.6.3.1 Kalman Gain The Schmidt–Kalman filter does not use the Kalman gain matrix. However, we need to write out its definition in partitioned form to show how its modification results in the Schmidt–Kalman gain.

The Kalman gain matrix would be partitionable conformably, such that

$$\begin{aligned} \overline{K} &= \left[\frac{K_\epsilon}{K_\nu}\right] \\ &= \left[\begin{array}{c|c} P_{\varepsilon\epsilon} & P_{\epsilon\nu} \\ \hline P_{\nu\varepsilon} & P_{\nu\nu} \end{array}\right] \left[\frac{H_\epsilon^{\mathrm{T}}}{H_\nu^{\mathrm{T}}}\right] \end{aligned} \tag{9.138}$$

$$\left\{[H_\epsilon \quad | \quad H_\nu] \left[\begin{array}{c|c} P_{\epsilon\epsilon} & P_{\epsilon\nu} \\ \hline P_{\nu\epsilon} & P_{\nu\nu} \end{array}\right] \left[\frac{H_\epsilon^{\mathrm{T}}}{H_\nu^{\mathrm{T}}}\right] + R\right\}^{-1} \tag{9.139}$$

and the individual blocks

$$K_\epsilon = \{P_{\epsilon\epsilon}H_\epsilon^{\mathrm{T}} + P_{\epsilon v}H_v^{\mathrm{T}}\}C, \tag{9.140}$$

$$K_v = \{P_{v\epsilon}H_\epsilon^{\mathrm{T}} + P_{vv}H_v^{\mathrm{T}}\}C, \tag{9.141}$$

where the common factor

$$C = \left\{ [H_\epsilon \quad | \quad H_v] \begin{bmatrix} P_{\epsilon\epsilon} & P_{\epsilon v} \\ P_{v\epsilon} & P_{vv} \end{bmatrix} \begin{bmatrix} H_\epsilon^{\mathrm{T}} \\ \hline H_v^{\mathrm{T}} \end{bmatrix} + R \right\}^{-1} \tag{9.142}$$

$$= \{H_\epsilon P_{\epsilon\epsilon}H_\epsilon^{\mathrm{T}} + H_\epsilon P_{\epsilon v}H_v^{\mathrm{T}} + H_v P_{v\epsilon}H_\epsilon^{\mathrm{T}} + H_v P_{vv}H_v^{\mathrm{T}} + R\}^{-1}. \tag{9.143}$$

However, the Schmidt–Kalman filter will, in effect, force K_v to be zero and redefine the upper block (no longer the K_ϵ of the Kalman filter) to be optimal under that constraint.

9.6.3.2 Suboptimal Approach The approach will be to define a suboptimal filter that does not estimate the nuisance state variables but does keep track of the influence they will have on the gains applied to the other state variables.

The suboptimal gain matrix for the Schmidt–Kalman filter has the form

$$\overline{K}_{\text{suboptimal}} = \begin{bmatrix} K_{\text{SK}} \\ 0 \end{bmatrix}, \tag{9.144}$$

where K_{SK} is the $n_\epsilon \times \ell$ Schmidt–Kalman gain matrix.

This suboptimal filter effectively ignores the nuisance states.

However, the calculation of the covariance matrix P used in defining the gain K_{SK} must still take into account the effect that this constraint has on the state estimation uncertainties and must optimize K_{SK} for that purpose. Here, K_{SK} will effectively be optimal for the constraint that the nuisance states are not estimated. However, the filter will still be *sub*optimal in the sense that filter performance using both Kalman gain blocks (K_ϵ and K_v) would be superior to that with K_{SK} alone.

The approach still propagates the full covariance matrix P, but the observational update equations are changed to reflect the fact that (in effect) $K_v = 0$.

9.6.3.3 Suboptimal Observational Update The observational update equation for arbitrary gain K_k can be represented in the form

$$P_{k(+)} = (I_n - \overline{K}_k H_k)P_{k(-)}(I_n - \overline{K}_k H_k)^{\mathrm{T}} + \overline{K}_k R \overline{K}_k^{\mathrm{T}}, \tag{9.145}$$

where n is the dimension of the state vector, I_n is the $n \times n$ identity matrix, ℓ is the dimension of the measurement, H_k is the $\ell \times n$ measurement sensitivity matrix, and R_k is the $\ell \times \ell$ covariance matrix of uncorrelated measurement noise.

In the case that the suboptimal gain K_k has the partitioned form shown in Equation 9.144, the partitioned observational update equation for P will be

$$\begin{aligned}\begin{bmatrix} P_{\epsilon\epsilon,k(+)} & P_{\epsilon\upsilon,k(+)} \\ P_{\upsilon\epsilon,k(+)} & P_{\upsilon\upsilon,k(+)} \end{bmatrix} &= \left(\begin{bmatrix} I_{n_\epsilon} & 0 \\ 0 & I_{n_\upsilon} \end{bmatrix} - \begin{bmatrix} K_{\mathrm{SK},k} \\ 0 \end{bmatrix} \begin{bmatrix} H_{\epsilon,k} & | & H_{\upsilon,k} \end{bmatrix} \right) \\ &\quad \times \begin{bmatrix} P_{\epsilon\epsilon,k(-)} & P_{\epsilon\upsilon,k(-)} \\ P_{\upsilon\epsilon,k(-)} & P_{\upsilon\upsilon,k(-)} \end{bmatrix} \\ &\quad \times \left(\begin{bmatrix} I_{n_\epsilon} & 0 \\ 0 & I_{n_\upsilon} \end{bmatrix} - \begin{bmatrix} K_{\mathrm{SK},k} \\ 0 \end{bmatrix} \begin{bmatrix} H_{\epsilon,k} & H_{\upsilon,k} \end{bmatrix} \right)^{\mathrm{T}} \\ &\quad + \begin{bmatrix} K_{\mathrm{SK},k} a \\ 0 \end{bmatrix} R_k \begin{bmatrix} K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \end{bmatrix}. \end{aligned} \tag{9.146}$$

The summed terms in parentheses can be combined into the following form for expansion:

$$\begin{aligned} &= \begin{bmatrix} I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k} & -K_{\mathrm{SK},k} H_{\upsilon,k} \\ 0 & I_{n_\upsilon} \end{bmatrix} \times \begin{bmatrix} P_{\epsilon\epsilon,k(-)} & P_{\epsilon\upsilon,k(-)} \\ P_{\upsilon\epsilon,k(-)} & P_{\upsilon\upsilon,k(-)} \end{bmatrix} \\ &\quad \times \begin{bmatrix} I_{n_\epsilon} - H_{\epsilon,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \\ -H_{\upsilon,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}} & I_{n_\upsilon} \end{bmatrix} + \begin{bmatrix} K_{\mathrm{SK},k} R_k K_{\mathrm{SK},k}^{\mathrm{T}} & 0 \\ 0 & 0 \end{bmatrix}, \end{aligned} \tag{9.147}$$

which can then be expanded to yield the following formulas for the blocks of P (with annotation showing intermediate results that can be reused to reduce the computation):

$$\begin{aligned} P_{\epsilon\epsilon,k(+)} &= \underbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})}_{\mathcal{A}} P_{\epsilon\epsilon,k(-)} \underbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})^{\mathrm{T}}}_{\mathcal{A}^{\mathrm{T}}} \\ &\quad - \underbrace{\overbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})}^{\mathcal{A}} P_{\upsilon\epsilon,k(-)} H_{\upsilon,k}^{\mathrm{T}} K_{\mathrm{SK},k}^{\mathrm{T}}}_{\mathcal{B}} \\ &\quad - \overbrace{K_{\mathrm{SK},k} H_{\upsilon,k} \; P_{\epsilon\upsilon,k(-)} (I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})^{\mathrm{T}}}^{\mathcal{B}^{\mathrm{T}}} \\ &\quad + K_{\mathrm{SK},k} R_k K_{\mathrm{SK},k}^{\mathrm{T}}, \end{aligned} \tag{9.148}$$

$$P_{\epsilon\upsilon,k(+)} = \overbrace{(I_{n_\epsilon} - K_{\mathrm{SK},k} H_{\epsilon,k})}^{\mathcal{A}} P_{\epsilon\upsilon,k(-)} - K_{\mathrm{SK},k} H_{\upsilon,k} P_{\upsilon\upsilon,k(-)}, \tag{9.149}$$

$$P_{\upsilon\epsilon,k(+)} = P_{\epsilon\upsilon,k(+)}^{\mathrm{T}}, \tag{9.150}$$

$$P_{\upsilon\upsilon,k(+)} = P_{\upsilon\upsilon,k(-)}. \tag{9.151}$$

TABLE 9.2 Implementation Equations of Schmidt–Kalman Filter

Observational Update

$$
\begin{aligned}
\mathcal{C} &= \{H_{\epsilon k}(P_{\epsilon\epsilon k(-)}H_{\epsilon k}^{\mathrm{T}} + P_{\epsilon v k(-)}H_{vk}^{\mathrm{T}}) \\
&\quad + H_{vk}(P_{v\epsilon k(-)}H_{\epsilon k}^{\mathrm{T}} + P_{vvk(-)}H_{vk}^{\mathrm{T}}) + R_k\}^{-1} \\
K_{\mathrm{SK},k} &= \{P_{\epsilon\epsilon k-}H_{\epsilon k}^{\mathrm{T}} + P_{\epsilon v k(-)}H_{vk}^{\mathrm{T}}\}\mathcal{C} \\
\mathrm{x}_{\epsilon,k(+)} &= \mathrm{x}_{\epsilon,k-} + K_{\mathrm{S-K},k}\{\mathrm{z}_k - H_{\epsilon k}\mathrm{x}_{\epsilon,k(-)}\} \\
\mathcal{A} &= I_{n_\epsilon} - K_{\mathrm{SK},k}H_{\epsilon,k} \\
\mathcal{B} &= \mathcal{A}P_{v\epsilon,k(-)}H_{v,k}^{\mathrm{T}}K_{S-K,k}^{\mathrm{T}} \\
P_{\epsilon\epsilon,k(+)} &= \mathcal{A}P_{\epsilon\epsilon,k-}\mathcal{A}^{\mathrm{T}} - \mathcal{B} - \mathcal{B}^{\mathrm{T}} + K_{\mathrm{SK},k}R_kK_{\mathrm{SK},k}^{\mathrm{T}} \\
P_{\epsilon v,k(+)} &= \mathcal{A}P_{\epsilon v,k(-)} - K_{\mathrm{SK},k}H_{v,k}P_{vv,k(-)} \\
P_{v\epsilon,k(+)} &= P_{\epsilon v,k(+)}^{\mathrm{T}} \\
P_{vv,k(+)} &= P_{vv,k(-)}
\end{aligned}
$$

Temporal Update

$$
\begin{aligned}
\mathrm{x}_{\epsilon,k+1(-)} &= \Phi_{\epsilon k}\mathrm{x}_{\epsilon,k(+)} \\
P_{\epsilon\epsilon k+1(-)} &= \Phi_{\epsilon k}P_{\epsilon\epsilon k(+)}\Phi_{\epsilon k}^{\mathrm{T}} + Q_{\epsilon\epsilon} \\
P_{\epsilon v k+1(-)} &= \Phi_{\epsilon k}P_{\epsilon v k(+)}\Phi_{vk}^{\mathrm{T}} \\
P_{v\epsilon k+1(-)} &= P_{\epsilon v k+1(-)}^{\mathrm{T}} \\
P_{vvk+1(-)} &= \Phi_{vk}P_{vvk(+)}\Phi_{vk}^{\mathrm{T}} + Q_{vv}
\end{aligned}
$$

Note that P_{vv} is unchanged by the observational update because $\mathbf{x}_v$ is not updated.

This completes the derivation of the Schmidt–Kalman filter. The temporal update of P in the Schmidt–Kalman filter will be the same as for the Kalman filter. This happens because the temporal update only models the propagation of the state variables, and the propagation model is the same in both cases.

9.6.4 Implementation Equations

We can now summarize just the essential equations from the derivation above, as listed in Table 9.2. These have been rearranged slightly to reuse intermediate results.

9.6.5 Computational Complexity

The purpose of the Schmidt–Kalman filter was to reduce the computational requirements over those required for the full Kalman filter. Although the equations appear to be more complicated, the dimensions of the matrices involved are smaller than the matrices in the Kalman filter.

We will now do a rough operations count of those implementation equations, just to be sure that they do, indeed, decrease computational requirements.

Table 9.3 is a breakdown of the operations counts for implementing the equations in Table 9.2. The formulas (in angular brackets) above the matrix formulas give the rough operations counts for implementing those formulas. An "operation" in this

accounting is roughly equivalent to a multiply and accumulate. The operations counts are expressed in terms of the number of measurements (ℓ, the dimension of the measurement vector), the number of essential state variables (n_ϵ), and the number of nuisance state variables (n_υ).

These complexity formulas are based on the matrix dimensions listed in Table 9.4.

A MATLAB implementation of the Schmidt–Kalman filter is in the m-file `KFvsSKF.m` on the Wiley web site.

9.7 MEMORY, THROUGHPUT, AND WORDLENGTH REQUIREMENTS

These may not be important issues for offline implementations of Kalman filters on mainframe scientific computers, but they can become critical issues for real-time implementations in embedded processors, especially as the dimensions of the state vector or measurement become larger. We present here some methods for assessing these requirements for a given application and for improving feasibility in marginal cases. These include order-of-magnitude plots of memory requirements and computational complexity as functions of the dimensions of the state vector and measurement vector. These plots cover the ranges from 1 to 1000 for these dimensions, which should include most problems of interest.

9.7.1 Wordlength Problems

9.7.1.1 Precision Problems Wordlength issues include precision problems (related to the number of significant bits in the mantissa field) and dynamic range problems (related to the number of bits in the exponent field). The issues and remedies related to precision are addressed in Chapter 7.

9.7.1.2 Scaling Problems Underflows and overflows are symptoms of dynamic range problems. These can often be corrected by rescaling the variables involved. This is equivalent to changing the units of measure, such as using kilometers in place of centimeters to represent length. In some cases, but not all cases, the condition number of a matrix can be improved by rescaling. For example, the two covariance matrices

$$\begin{bmatrix} 1 & 0 \\ 0 & \epsilon^2 \end{bmatrix} \text{ and } \frac{1}{2}\begin{bmatrix} 1+\epsilon^2 & 1-\epsilon^2 \\ 1-\epsilon^2 & 1+\epsilon^2 \end{bmatrix}$$

have the same condition number ($1/\epsilon^2$), which can be troublesome for very small values of ϵ. The condition number of the matrix on the left can be made equal to 1 by simply rescaling the second component by $1/\epsilon$.

9.7.2 Memory Requirements

In the early years of Kalman filter implementation, a byte of memory cost about as much as a labor hour at minimum wage. With these economic constraints, programmers developed many techniques for reducing the memory requirements of Kalman

TABLE 9.3 Operations Counts for the Schmidt–Kalman Filter

Scalar Operation Counts for Matrix Operations	Totals by Rows
$\mathcal{C} = \{\overbrace{H_{\epsilon k}}^{\langle n_\epsilon \ell^2\rangle} \times (\underbrace{\overbrace{P_{\epsilon\epsilon k(-)}}^{\langle n_\epsilon^2 \ell\rangle} H_{\epsilon k}^{\mathrm{T}} + \overbrace{P_{\epsilon v k(-)}}^{\langle n_\epsilon n_v \ell\rangle} H_{vk}^{\mathrm{T}}}_{\langle \text{used again below}\rangle})$	$n_\epsilon \ell^2 + n_\epsilon^2 \ell + n_\epsilon n_v \ell$
$+ \overbrace{H_{vk}}^{\langle n_v \ell^2\rangle} \times (\overbrace{P_{v\epsilon k(-)}}^{\langle n_\epsilon n_v \ell\rangle} H_{\epsilon k}^{\mathrm{T}} + \overbrace{P_{vvk(-)}}^{\langle n_v^2 \ell\rangle} H_{vk}^{\mathrm{T}})$	$n_v \ell^2 + n_\epsilon n_v \ell + n_v^2 \ell$
$+R_k\}^{-1\langle \ell^3\rangle}$(matrix inverse)	ℓ^3
$K_{SK,k} = \{\overbrace{P_{\epsilon\epsilon k(-)} H_{\epsilon k}^{\mathrm{T}} + P_{\epsilon v k(-)} H_{vk}^{\mathrm{T}}}^{\langle \text{already computed above}\rangle}\} \overbrace{\times \mathcal{C}}^{\langle n_\epsilon \ell^2\rangle}$	$n_\epsilon \ell^2$
$\mathrm{x}_{\epsilon,k(+)} = \mathrm{x}_{\epsilon,k(-)} + \overbrace{K_{\mathrm{SK},k}}^{\langle n_\epsilon \ell\rangle} \times \overbrace{\{\mathrm{z}_k - H_{\epsilon k}\mathrm{x}_{\epsilon,k(-)}\}}^{\langle n_\epsilon \ell\rangle}$	$2n_\epsilon \ell$
$\mathcal{A} = I_{n_\epsilon} - \overbrace{K_{\mathrm{SK},k}}^{\langle n_\epsilon^2 \ell\rangle} H_{\epsilon,k}$	$n_\epsilon^2 \ell$
$\mathcal{B} = \underbrace{\overbrace{\mathcal{A}}^{\langle n_\epsilon^2 n_v\rangle} \times P_{\epsilon v,k(-)}}_{\langle n_\epsilon^2 n_v\rangle} \times \overbrace{H_{v,k}^{\mathrm{T}} \times K_{\mathrm{SK},k}^{\mathrm{T}}}^{\langle n_\epsilon n_v \ell\rangle}$	$2n_\epsilon^2 n_v + n_\epsilon n_v \ell$
$P_{\epsilon\epsilon,k(+)} = \mathcal{A} P_{\epsilon\epsilon,k(-)} \mathcal{A}^{\mathrm{T}} - \mathcal{B} - \mathcal{B}^{\mathrm{T}}$	$\frac{3}{2}n_\epsilon^3 + \frac{1}{2}n_\epsilon^2$
$+\underbrace{K_{\mathrm{SK},k}\overbrace{[H_{vk}, P_{vv,k(-)} H_{v,k}^{\mathrm{T}} + R_k]}^{\langle n_\epsilon^2 n_v\rangle} K_{\mathrm{SK},k}^{\mathrm{T}}}_{\langle \frac{1}{2}n_\epsilon^2 \ell + n_\epsilon \ell^2 + \frac{1}{2}n_\epsilon \ell\rangle}$	$\frac{3}{2}n_v^3 + n_v^2 \ell + \frac{1}{2}n_v \ell^2 + \frac{1}{2}n_v^2 + \frac{1}{2}n_v \ell$
$P_{\epsilon v,k(+)} = \overbrace{\mathcal{A} \times P_{\epsilon v,k(-)}}^{\langle n_\epsilon^2 n_v\rangle} - \overbrace{K_{\mathrm{SK},k} \times H_{v,k} \times P_{vv,k(-)}}^{\langle n_\epsilon n_v \ell\rangle \quad \langle n_\epsilon n_v^2\rangle}$	$n_\epsilon^2 n_v + n_\epsilon n_v \ell + n_\epsilon n_v^2$
$P_{ve,k(+)} = P_{\epsilon v,k(+)}^{\mathrm{T}}$	0
$P_{vv,k(+)} = P_{vv,k(-)}$	0
Total for Observational Update	$3n_\epsilon \ell^2 + \frac{5}{2}n_\epsilon^2 \ell + 4n_\epsilon n_v \ell + \frac{3}{2}n_v \ell^2 + 2n_v^2 \ell + \ell^3 + \frac{5}{2}n_\epsilon \ell + 3n_\epsilon^2 n_v + \frac{1}{2}n_v \ell + n_\epsilon n_v^2$
Temporal Update	
$\hat{\mathbf{x}}_{\epsilon,k+1(-)} = \Phi_{\epsilon k} \hat{\mathbf{x}}_{\epsilon,k(+)}$	n_ϵ^2
$P_{\epsilon\epsilon k+1(-)} = \Phi_{\epsilon k} P_{\epsilon\epsilon k(+)} \Phi_{\epsilon k}^{\mathrm{T}} + Q_{\epsilon\epsilon}$	$\frac{3}{2}n_\epsilon^3 + \frac{1}{2}n_\epsilon^2$
$P_{\epsilon v k+1(-)} = \Phi_{\epsilon k} P_{\epsilon v k(+)} \Phi_{vk}^{\mathrm{T}}$	$n_\epsilon n_v^2 + n_v n_\epsilon^2$
$P_{ve k+1(-)} = P_{\epsilon v k+1-}^{\mathrm{T}}$	0
$P_{vv k+1(-)} = \Phi_{vk} P_{vv k(+)} \Phi_{vk}^{\mathrm{T}} + Q_{vv}$	$\frac{3}{2}n_v^3 + \frac{1}{2}n_v^2$
Total for Temporal Update	$\frac{3}{2}n_\epsilon^2 + \frac{3}{2}n_\epsilon^3 + n_\epsilon n_v^2 + n_\epsilon^2 n_v + \frac{3}{2}n_v^3 + \frac{1}{2}n_v^2$
Total for Schmidt–Kalman Filter	$3n_\epsilon \ell^2 + \frac{5}{2}n_\epsilon^2 \ell + 4n_\epsilon n_v \ell + \frac{3}{2}n_v \ell^2 + 2n_v^2 \ell + \ell^3 + \frac{5}{2}n_\epsilon \ell + 4n_\epsilon^2 n_v + \frac{3}{2}n_\epsilon^3 + \frac{3}{2}n_\epsilon^2 + \frac{1}{2}n_v \ell + \frac{3}{2}n_v^3 + \frac{1}{2}n_v^2 + \frac{1}{2}n_v \ell + 2n_\epsilon n_v^2$

TABLE 9.4 Array Dimensions

Symbol	Rows	Columns
$\mathcal{A}$	n_ϵ	n_ϵ
$\mathcal{B}$	n_ϵ	n_ϵ
$\mathcal{C}$	ℓ	ℓ
H_ϵ	ℓ	n_ϵ
H_v	ℓ	n_v
K_{SK}	n_ϵ	ℓ
$P_{\epsilon\epsilon}$	n_ϵ	n_ϵ
$P_{\epsilon v}$	n_ϵ	n_v
$P_{v\epsilon}$	n_v	n_ϵ
P_{vv}	n_v	n_v
$Q_{\epsilon\epsilon}$	n_ϵ	n_ϵ
Q_{vv}	n_v	n_v
R	ℓ	ℓ
Φ_ϵ	n_ϵ	n_ϵ
Φ_v	n_v	n_v
$\hat{\mathbf{x}}_\epsilon$	n_ϵ	1
$\mathbf{z}$	ℓ	1

filters. A few of these techniques have been mentioned in Chapter 7, although they are not as important as they once were. Memory costs have dropped dramatically since these methods were developed. The principal reason for paying attention to memory requirements nowadays is to determine the limits on problem size with a fixed memory allocation. Memory is cheap, but still finite.

9.7.2.1 Program Memory versus Data Memory In the "von Neumann architecture" for processing systems, there is no distinction between the memory containing the algorithms and that containing the data used by the algorithms. In specific applications, the program may include formulas for calculating the elements of arrays such as Φ or H. Other than that, the memory requirements for the algorithms tend to be independent of application and the "problem size." For the Kalman filter, the problem size is specified by the dimensions of the state (n), measurement (ℓ), and process noise (p). The data memory requirements for storing the arrays with these dimensions is very much dependent on the problem size. We present here some general formulas for this dependence.

9.7.2.2 Data Memory and Wordlength The data memory requirements will depend upon the data wordlengths (in bits) as well as the size of the data structures. The data requirements are quoted in "floating-point words." These are either 4- or 8-byte words (in IEEE floating-point standard formats) for the examples presented in this book.

TABLE 9.5 Array Requirements for "Conventional" Kalman Filter Implementation

Functional Grouping	Matrix Expression	Array Dimensions	Total Memory*
Riccati equation	P	$n \times n$	
	ΦP	$n \times n$	
	GQG^{T}	$n \times n$	$3n^2$
	HP PH^{T}	$\ell \times n$	$+\ell n$
	R	$\ell \times \ell$	
	$HPH^{\mathrm{T}} + R$ $[HPH^{\mathrm{T}} + R]^{-1}$	$\ell \times \ell$	$+2\ell^2$
Common	Φ	$n \times n$	n^2
	H	$\ell \times n$	
	$\overline{K}$	$n \times \ell$	$+2\ell n$
Linear estimation	z $z - Hx$	ℓ	ℓ
	x	n	
	Φx	n	$+2n$

*In units of floating-point data words.

9.7.2.3 Data Memory Requirements These are also influenced somewhat by programming style, particularly by the ways that data structures containing partial results are reused.

The data memory requirements for a more-or-less "conventional" implementation of the Kalman filter are listed in Table 9.5 and plotted in Figure 9.26. This is the Kalman filter implementation diagrammed in Figure 7.2, which reuses some partial results. The array dimensions are associated with the results of matrix subexpressions that they contain. These are divided into three groups:

1. those arrays common to both the Riccati equation (for covariance and gain computations) and the linear estimation computations;
2. the additional array expressions required for solving the Riccati equation, which provides the Kalman gain as a partial result; and
3. the additional array expressions required for linear estimation of the state variables, given the Kalman gains.

Expressions grouped together by curly braces are assumed to be kept in the same data structures. This implementation assumes that

- the product GQG^{T} is input and not computed (which eliminates any dependence on p, the dimension of Q);
- given $\Phi P, \Phi$, and GQG^{T}, the operation $P \leftarrow \Phi P \Phi^{\mathrm{T}} + GQG^{\mathrm{T}}$ is performed in place;

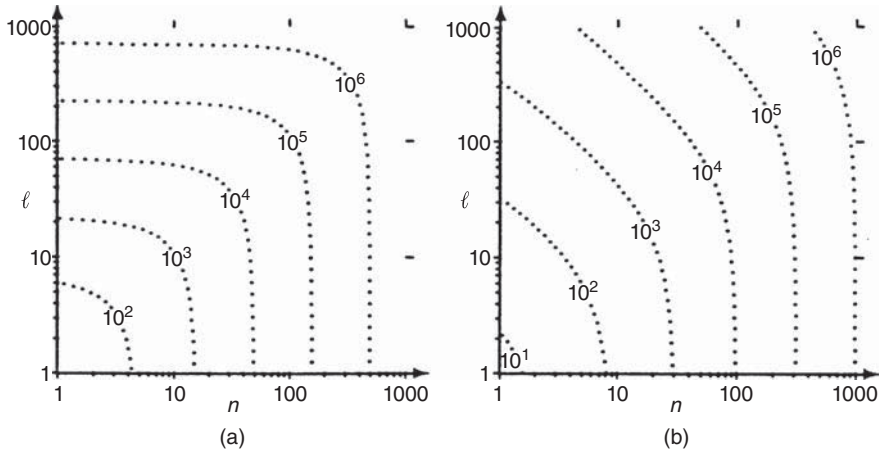

Figure 9.26 Conventional filter memory requirements (in words) versus state dimension (n) and measurement dimension (l). (a) Complete implementation and (b) without gain calculations.

- computations involving the subexpression PH^{T} can be implemented with HP by changing the indexing;
- $HPH^{\mathrm{T}} + R$ can be computed in place (from HP, H, and R) and inverted in place;
- $z - Hx$ can be computed in place (in the z array); and
- the state update computation $x \leftarrow (\Phi x) + \overline{K}[z - H(\Phi x)]$ requires additional memory only for the intermediate result (Φx).

Figure 9.26 illustrates the numerical advantage of precomputing and storing the Kalman gains in bulk memory. It saves about a factor of 4 in data memory requirements for small dimensions of the measurement relative to the state and even more for larger measurement dimensions.

9.7.2.4 Eliminating Data Redundancy Reuse of temporary arrays is not the only way to save memory requirements. It is also possible to save data memory by eliminating redundancies in data structures. The symmetry of covariance matrices is an example of this type of redundancy. The methods discussed here depend on such constraints on matrices that can be exploited in designing their data structures. They do require additional programming effort, however, and the resulting runtime program code may require slightly more memory and more processing time. The difference will be primarily from index computations, which are not the standard ones used by optimizing compilers. Table 9.6 lists some common constraints on square matrices, the minimum memory requirements (as multiples of the memory required for a scalar variable), and corresponding indexing schemes for packing the matrices in singly subscripted arrays. The indexing schemes are given as formulas for the single subscript (k) corresponding to the row (i) and column (j) indices of a two-dimensional array.

TABLE 9.6 Minimum Memory Requirements for $n \times n$ Matrices*

Matrix Type	Minimum Memory†	Indexing $k(i, j)$
Symmetric	$\frac{n(n+1)}{2}$	$i + \frac{j(j-1)}{2}$ $\frac{(2n-i)(i-1)}{2} + j$
Upper triangular	$\frac{n(n+1)}{2}$	$i + \frac{j(j-1)}{2}$
Unit upper triangular	$\frac{n(n+1)}{2}$	$i + \frac{(j-1)(j-2)}{2}$
Strictly upper triangular	$\frac{n(n-1)}{2}$	$i + \frac{(j-1)(j-2)}{2}$
Diagonal	n	i
Toeplitz	n	$i+j-1$

*Note: n is the dimension of the matrix; i and j are the indices of a two-dimensional array; and k is the corresponding index of a one-dimensional array.
†In units of data words.

The two formulas given for symmetric matrices correspond to the two alternative indexing schemes:

$$\begin{bmatrix} 1 & 2 & 4 & \dots & \frac{1}{2}n(n-1)+1 \\ & 3 & 5 & \dots & \frac{1}{2}n(n-1)+2 \\ & & 6 & \dots & \frac{1}{2}n(n-1)+3 \\ & & & \ddots & \vdots \\ & & & \dots & \frac{1}{2}n(n+1) \end{bmatrix} \text{or} \begin{bmatrix} 1 & 2 & 3 & \dots & n \\ & n+1 & n+2 & \dots & 2n-1 \\ & & 2n & \dots & 3n-3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ & & & \dots & \frac{1}{2}n(n+1) \end{bmatrix},$$

where the element in the ith row and jth column is $k(i, j)$.

Just by exploiting symmetry or triangularity, these methods can save about a factor of 2 in memory with a fixed state vector dimension or allow about a factor of $\sqrt{2}$ increase in the state vector dimension (about 40% increase in the dimension) with the same amount of memory.

Arrays Can Be Replaced by Algorithms In special cases, data arrays can be eliminated entirely by using an algorithm to compute the matrix elements "on the fly." For example, the companion form coefficient matrix (F) and the corresponding state-transition matrix (Φ) for the differential operator d^n/dt^n are

$$F = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{9.152}$$

$$\Phi(t) = e^{Ft} \tag{9.153}$$

$$= \begin{bmatrix} 1 & t & \frac{1}{2}t^2 & \cdots & \frac{1}{(n-1)!}t^{n-1} \\ 0 & 1 & t & \cdots & \frac{1}{(n-2)!}t^{n-2} \\ 0 & 0 & 1 & \cdots & \frac{1}{(n-3)!}t^{n-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{9.154}$$

where t is the discrete-time interval. The following algorithm computes the product $M = \Phi P$, with Φ as given in Equation 9.154, using only P and t:

```
for i = 1: n,
    for j = 1: n,
        s = P (n, j);
        m = n - 1;
        for k = n - 1: 1: i,
            s = P (k, j) + s * t/m;
            m = m - 1;
        end;
        M (i, j) = s;
    end;
end;
```

It requires about half as many arithmetic operations as a general matrix multiply and requires no memory allocation for one of its matrix factors.

9.7.3 Throughput, Processor Speed, and Computational Complexity

9.7.3.1 Computational Complexity and Throughput The "throughput" of a Kalman filter implementation is related to how many updates it can perform in a unit of time. This depends upon the speed of the host processor, in floating-point operations (flops) per second, and the computational complexity of the application, in flops per filter update:

$$\text{Throughput}\left(\frac{\text{Updates}}{\text{s}}\right) = \frac{\text{Processor speed (flops/s)}}{\text{Computational complexity (flops/update)}}.$$

The numerator of the expression on the right-hand side depends upon the host processor. Formulas for the denominator, as functions of the application problem size, are derived in Chapter 7. These are the *maximum* computational complexities of the implementation methods listed in Chapter 7. The computational complexity of an application can be made smaller if it includes sparse matrix operations that can be implemented more efficiently.

9.7.3.2 Conventional Kalman Filter The maximum computational complexity of the conventional Kalman filter implementation is plotted versus problem size in Figure 9.27. This implementation uses all the shortcuts listed in Section 9.7.2 and also

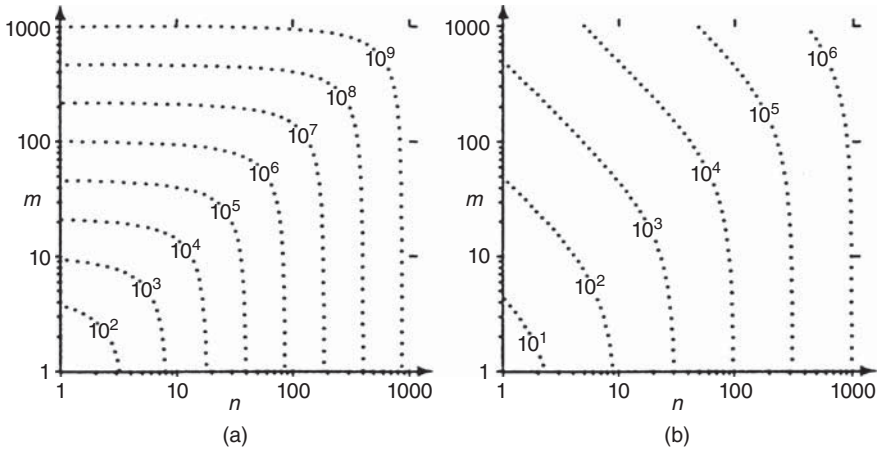

Figure 9.27 Contour plots of computational complexity (in flops per measurement) of the Kalman filter as a function of state dimension (n) and measurement dimension (m). (a) Complete and (b) without the Riccati equation.

eliminates redundant computations in symmetric matrix products. The right-hand plot assumes that the matrix Riccati equation for the Kalman gain computations has been solved offline, as in a gain-scheduled or steady-state implementation.

9.7.3.3 Bierman–Thornton Square-Root Implementation The corresponding dependence of computational complexity on problem size for the *UD* filter implementation is plotted in Figure 9.28(a). These data include the computational cost of diagonalizing Q and R on each temporal and observational update, respectively. The corresponding results for the case that Q and R are already diagonal are displayed in Figure 9.28(b).

9.7.4 Programming Cost versus Runtime Cost

The computational complexity issue in Kalman filtering is usually driven by the need to execute in real time. The computational complexity grows so fast with the problem size that it will overwhelm even the fastest processors for sufficiently large system models. For that reason, the issue of computational complexity is one that must be addressed early on in the filter design cycle.

Another trade-off in the design of Kalman filters is between the one-time cost of programming the implementation and the recurring cost of running it on a computer. As computers grow less expensive compared to programmers, this trade-off tends to favor the most straightforward methods, even those that cause numerical analysts to wince. Keep in mind, however, that this is a low cost/high risk approach. Remember that the reason for the development of better implementation methods was the failure of the straightforward programming solutions to produce acceptable results.

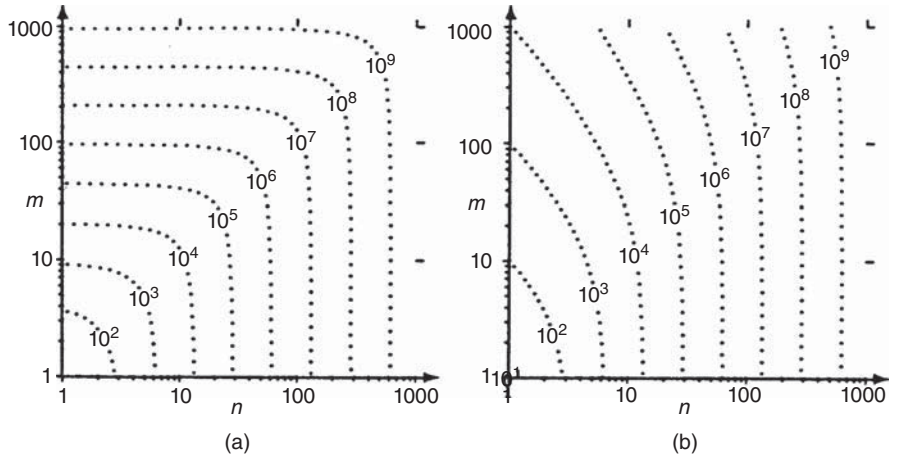

Figure 9.28 Contour plots of computational complexity (in flops per measurement) of the Bierman–Thornton implementation as a function of state dimension (n) and measurement dimension (m). (a) Full Q and R matrices and (b) diagonal Q and R matrices.

9.8 WAYS TO REDUCE COMPUTATIONAL REQUIREMENTS

9.8.1 Reducing Complexities of Matrix Products

9.8.1.1 Products of Two Matrices The number of flops required to compute the product of general $\ell \times m$ and $m \times n$ matrices is $\ell m^2 n$. This figure can be reduced substantially for matrices with predictable patterns of zeros or symmetry properties. These tricks can be used to advantage in computing matrix products that are symmetric and for products involving diagonal or triangular factors. They should always be exploited whenever H or Φ is a sparse matrix.

9.8.1.2 Products of Three Matrices It is of considerable practical importance that *associativity of matrix multiplication does not imply invariance of computational complexity*. The associativity of matrix multiplication is the property that

$$M_1 \times (M_2 \times M_3) = (M_1 \times M_2) \times M_3 \tag{9.155}$$

for conformably dimensioned matrices M_1, M_2, and M_3. That is, the *result* is guaranteed to be independent of the *order* in which the two matrix multiplications are performed. However, the *effort* required to obtain the result is *not* always independent of the order of multiplication. This distinction is evident if one assigns conformable dimensions to the matrices involved and evaluates the number of scalar multiplications required to compute the result, as shown in Table 9.7. The number of flops depends on the order of multiplication, being $n_2(n_3^3 + n_1 n_2)n_4$ in one case and $n_1(n_2^2 + n_3 n_4)n_3$ in the other case. The implementation $M_1 \times (M_2 \times M_3)$ is favored if

TABLE 9.7 Computational Complexities of Triple Matrix Product

Attribute	Value	
Implementation	$\underbrace{M_1}_{n_1 \times n_2} \times (\underbrace{M_2}_{n_2 \times n_3} \times \underbrace{M_3}_{n_3 \times n_4})$	$(\underbrace{M_1}_{n_1 \times n_2} \times \underbrace{M_2}_{n_2 \times n_3}) \times \underbrace{M_3}_{n_3 \times n_4}$
Number of flops first multiply	$n_2 n_3^2 n_4$	$n_1 n_2^2 n_3$
Second multiply	$n_1 n_2^2 n_4$	$n_1 n_3^2 n_4$
Total	$n_2(n_3^3 + n_1 n_2)n_4$	$n_1(n_2^2 + n_3 n_4)n_3$

$n_1 n_2^2(n_4 - n_3) < (n_1 - n_2)n_3^2 n_4$, and the implementation $(M_1 \times M_2) \times M_3$ is favored if the inequality is reversed. The correct selection is used to advantage in the more practical implementations of the Kalman filter, such as the De Vries implementation (see Section 7.6.1.4).

9.8.2 Offline versus Online Computational Requirements

The Kalman filter is a "real-time" algorithm, in the sense that it calculates an estimate of the current state of a system given measurements obtained in real time. In order that the filter be implementable in real time, however, it must be possible to execute the algorithm in real time with the available computational resources. In this assessment, it is important to distinguish between those parts of the filter algorithm that must be performed "online" and those that can be performed "offline" (i.e., carried out beforehand, with the results stored in memory, including bulk media, such as magnetic tape or CDROM, and read back in real time[7]). The online computations are those that depend upon the measurements of the real-time system. Those calculations cannot be made until their input data become available.

It is noted in Chapter 5 that the computations required for calculating the Kalman gains do not depend upon the real-time data, and for that reason, they can be executed offline. It is repeated here for emphasis and to formalize some of the practical methods used for implementation.

The most straightforward method is to precompute the gains and store them for retrieval in real time. This is also the method with the most general applicability. Some methods of greater efficiency (but less generality) are discussed in the following subsections. Methods for performance analysis of these suboptimal estimation methods are discussed in Section 9.5.

9.8.3 Gain Scheduling

This is an approximation method for estimation problems in which the rate of change of the Kalman gains is very slow compared to the sampling rate. Typically, the relative

[7] In assessing the real-time implementation requirements, one must trade off the time to read these prestored values versus the time required to compute them. In some cases, the read times may exceed the computation times.

change in the Kalman gain between observation times may be a few percent or less. In that case, one value of the Kalman gain may be used for several observation times. Each gain value is used for a "stage" of the filtering process.

This approach is typically used for problems with constant coefficients. The gains in this case have an asymptotic constant value but go through an initial transient due to larger or smaller initial uncertainties than the steady-state uncertainties. A few "staged" values of the gains during that transient phase may be sufficient to achieve adequate performance. The values used may be sampled values in the interior of the stage in which they are used or weighted averages of all the exact values over the range.

The performance trade-off between the decreased storage requirements (for using fewer values of the gains) and the increased approximation error (due to differences between the optimal gains and the scheduled gains) can be analyzed by simulation.

9.8.4 Steady-State Gains for Time-Invariant Systems

This is the limiting case of gain scheduling—with only one stage—and it is one of the more common uses of the algebraic Riccati equation. In this case, only the asymptotic values of the gains are used. This requires the solution of the algebraic (steady-state) matrix Riccati equation.

There are several methods for solving the steady-state matrix Riccati equation in the following subsections. One of these (the doubling method) is based on the linearization method for the Riccati equation presented in Chapter 5. Theoretically, it converges exponentially faster than the serial iteration method. In practice, however, convergence can stall (due to numerical problems) before an accurate solution is attained. However, it can still be used to obtain a good starting estimate for the Newton–Raphson method (described in Chapter 5).

9.8.4.1 Doubling Method for Time-Invariant Systems This is an iterative method for approximating the asymptotic solution to the *time-invariant* Riccati equation, based on the formula given in Lemma 5.2. As in the continuous case, the asymptotic solution should equal the solution of the *steady-state equation*:

$$P_\infty = \Phi[P_\infty - P_\infty H^{\mathrm{T}}(HP_\infty H^{\mathrm{T}} + R)^{-1} HP_\infty]\Phi^{\mathrm{T}} + Q, \tag{9.156}$$

although this is not the form of the equation that is used. Doubling methods generate the sequence of solutions

$$P_{1(-)}, P_{2(-)}, P_{4(-)}, P_{8(-)}, \ \ldots\ , P_{2^k(-)}, P_{2^{k+1}(-)}, \ \ldots$$

of the nonalgebraic matrix Riccati equation as an initial-value problem—by doubling the time interval between successive solutions. The doubling speedup is achieved by successive squaring of the equivalent state-transition matrix for the time-invariant Hamiltonian matrix

$$\Psi = \begin{bmatrix} (\Phi + Q\Phi^{-\mathrm{T}} HR^{-1}H^{\mathrm{T}}) & Q\Phi^{-\mathrm{T}} \\ \Phi^{-\mathrm{T}} R^{-1} & \Phi^{-\mathrm{T}} \end{bmatrix}. \tag{9.157}$$

The pth squaring of Ψ will then yield Ψ^{2p} and the solution

$$P_{2p(-)} = A_{2p}B_{2p}^{-1} \tag{9.158}$$

for

$$\begin{bmatrix} A_{2p} \\ B_{2p} \end{bmatrix} = \Psi^{2p} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix}. \tag{9.159}$$

Davison–Maki–Friedlander–Kailath Squaring Algorithm Note that if one expresses Ψ^{2^N} in symbolic form as

$$\Psi^{2^N} = \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + \mathcal{C}_N\mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{C}_N\mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix}, \tag{9.160}$$

then its square can be put in the form

$$\Psi^{2^{N+1}} = \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + \mathcal{C}_N\mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{C}_N\mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix}^2 \tag{9.161}$$

$$= \begin{bmatrix} \mathcal{A}_{N+1}^{\mathrm{T}} + \mathcal{C}_{N+1}\mathcal{A}_{N+1}^{-1}\mathcal{B}_{N+1} & \mathcal{C}_{N+1}\mathcal{A}_{N+1}^{-1} \\ \mathcal{A}_{N+1}^{-1}\mathcal{B}_{N+1} & \mathcal{A}_{N+1}^{-1} \end{bmatrix}, \tag{9.162}$$

$$\mathcal{A}_{N+1} = \mathcal{A}_N(I + \mathcal{B}_N\mathcal{C}_N)^{-1}\mathcal{A}_N, \tag{9.163}$$

$$\mathcal{B}_{N+1} = \mathcal{B}_N + \mathcal{A}_N(I + \mathcal{B}_N\mathcal{C}_N)^{-1}\mathcal{B}_N\mathcal{A}_N^{\mathrm{T}}, \tag{9.164}$$

$$\mathcal{C}_{N+1} = \mathcal{C}_N + \mathcal{A}_N^{\mathrm{T}}\mathcal{C}_N(I + \mathcal{B}_N\mathcal{C}_N)^{-1}\mathcal{A}_N. \tag{9.165}$$

The last three equations define an algorithm for squaring Ψ^{2^N}, starting with the values of $\mathcal{A}_N$, $\mathcal{B}_N$, and $\mathcal{C}_N$ for $N = 0$, given by Equation 9.157:

$$\mathcal{A}_0 = \Phi^{\mathrm{T}}, \tag{9.166}$$

$$\mathcal{B}_0 = H^{\mathrm{T}}R^{-1}H, \tag{9.167}$$

$$\mathcal{C}_0 = Q. \tag{9.168}$$

9.8.4.2 Initial Conditions If the initial value of the Riccati equation is with $P_0 = 0$, the zero matrix, it can be represented by $P_0 = A_0B_0^{-1}$ for $A_0 = 0$ and any nonsingular

TABLE 9.8 Davison–Maki–Friedlander–Kailath Squaring Algorithm

Initialization	Iteration (N times)
$\mathcal{A} = \Phi^{\mathrm{T}}$	$\mathcal{A} \leftarrow \mathcal{A}(I + \mathcal{B}C)^{-1}\mathcal{A}^{\mathrm{T}}$
$\mathcal{B} = H^{\mathrm{T}}R^{-1}H$	$\mathcal{B} \leftarrow \mathcal{B} + \mathcal{A}(I + \mathcal{B}C)^{-1}\mathcal{B}\mathcal{A}^{\mathrm{T}}$
$C = Q \ = P_1$	$C \leftarrow C + \mathcal{A}^{\mathrm{T}}C(I + \mathcal{B}C)^{-1}\mathcal{A}$
Termination	
$P_{2^N} = C$	

B_0. Then the Nth iterate of the doubling algorithm will yield

$$\begin{bmatrix} A_{2^N} \\ B_{2^N} \end{bmatrix} = \Psi^{2^N} \begin{bmatrix} A_0 \\ B_0 \end{bmatrix} \tag{9.169}$$

$$= \begin{bmatrix} \mathcal{A}_N^{\mathrm{T}} + C_N\mathcal{A}_N^{-1}\mathcal{B}_N & C_N\mathcal{A}_N^{-1} \\ \mathcal{A}_N^{-1}\mathcal{B}_N & \mathcal{A}_N^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ B_1 \end{bmatrix} \tag{9.170}$$

$$= \begin{bmatrix} C_N\mathcal{A}_N^{-1}\mathcal{B}_1 \\ \mathcal{A}_N^{-1}B_1 \end{bmatrix}, \tag{9.171}$$

$$P_{2^N} = A_{2^N}B_{2^N}^{-1} \tag{9.172}$$

$$= C_N\mathcal{A}_N^{-1}B_1(\mathcal{A}_N^{-1}B_1)^{-1} \tag{9.173}$$

$$= C_N. \tag{9.174}$$

That is, after the Nth squaring step, the submatrix

$$C_N = P_{2^N}. \tag{9.175}$$

The resulting algorithm is summarized in Table 9.8. It has complexity $\mathcal{O}(n^3 \log\ k)$ flops for computing P_k, requiring one $n \times n$ matrix inverse and eight $n \times n$ matrix products per iteration[8]. An array allocation scheme for performing the squaring algorithm using only six $n \times n$ arrays is shown in Figure 9.29.

9.8.4.3 Numerical Convergence Problems Convergence can be stalled by precision limitations before it is complete. The problem is that the matrix $\mathcal{A}$ is effectively squared on each iteration and appears quadratically in the update equations for $\mathcal{B}$ and C. Consequently, if $\|\mathcal{A}_N\| \ll 1$, then the computed values of $\mathcal{B}_N$ and C_N may become stalled numerically as $\|\mathcal{A}_N\| \to 0$ exponentially. The value of $\mathcal{A}_N$ can be monitored to test for this stall condition. Even in those stall situations, the doubling algorithm is still an efficient method for getting an approximate nonnegative-definite solution.

[8]The matrices $\mathcal{B}_N$ and C_N and the matrix products $(I + \mathcal{B}_N C_N)^{-1}B_N$ and $C_N(I + \mathcal{B}_N C_N)^{-1}$ are symmetric. That fact can be exploited to eliminate $2n^2(n-1)$ flops per iteration. With these savings, this algorithm requires slightly fewer flops per iteration than the straightforward squaring method. It requires about one-fourth less memory than straightforward squaring, also.

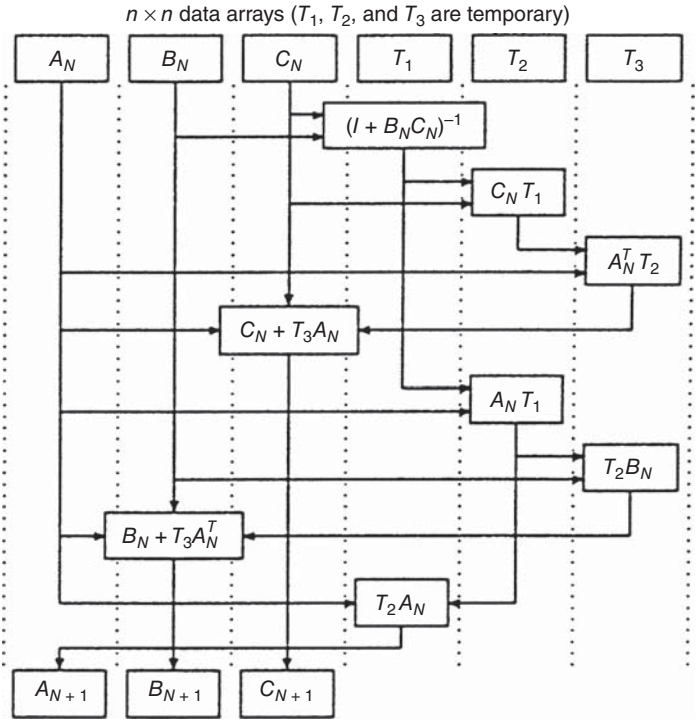

Figure 9.29 Data array usage for doubling algorithm.

9.9 ERROR BUDGETS AND SENSITIVITY ANALYSIS

9.9.1 Design Problem for Statistical Performance Requirements

This is the problem of estimating the statistical performance of a sensor system that will make measurements of some dynamic and stochastic process and estimate its state. Statistical performance is defined by mean-squared errors at the "system level"; these depend on mean-squared errors at the subsystem level; and so on down to the level of individual sensors and components. The objective of this activity is to be able to justify the apportionment of these lower level performance requirements.

This type of performance analysis is typically performed during the preliminary design of estimation systems. The objective of the analysis is to evaluate the feasibility of an estimation system design for meeting some prespecified acceptable level of uncertainty in the estimates that will be obtained.

The Kalman filter does not design sensor systems, but it provides the tool for doing it defensibly. That tool is the model for estimation uncertainty. The covariance propagation equations derived from the model can be used in characterizing estimation uncertainty as a function of the "parameters" of the design. Some of these parameters are statistical, such as the noise models of the sensors under consideration. Others are

deterministic. The deterministic parameters may also be discrete valued—such as the sensor type—or continuous—such as the sensor location.

One of the major uses of Kalman filtering theory is in the design of sensor systems:

1. Vehicle navigation systems containing some combination of sensors, such as
   (a) Attitude and attitude rate sensors
      i. Magnetic compass (field sensor)
      ii. Displacement gyroscopes
      iii. Star trackers or sextants
      iv. Rate gyroscopes
      v. Electric field sensors (for earth potential field)
   (b) Acceleration sensors (accelerometers)
   (c) Velocity sensors (e.g., onboard Doppler radar)
   (d) Position sensors
      i. Global Navigation Satellite System (GNSS)
      ii. Terrain-mapping radar
      iii. Long-range navigation (LORAN)
      iv. Instrument Landing System (ILS)
2. Surface-based, airborne, or spaceborne tracking systems
   (a) Range and Doppler radar
   (b) Imaging sensors (e.g., visible or infrared cameras).

In the design of these systems, it is assumed that a Kalman filter will be used in estimating the dynamic state (position and velocity) of the vehicle. Therefore, the associated covariance equations can be used to estimate the performance in terms of the covariance of estimation uncertainty.

9.9.2 Error Budgeting

Large systems such as spacecraft and aircraft contain many sensors of many types, and the Kalman filter provides a methodology for the integrated design of such systems. Error budgeting is a specialized form of sensitivity analysis. It uses the error covariance equations of the Kalman filter to formalize the dependence of system accuracy on the component accuracies of its individual sensors. This form of covariance analysis is significantly more efficient than Monte Carlo analysis for this purpose, although it does depend upon linearity of the underlying dynamic processes.

Error budgeting is a process for trading off performance requirements among sensors and subsystems of a larger system for meeting a diverse set of overall performance constraints imposed at the "system level." The discussion here is limited to the system-level requirements related to *accuracy*, although most system requirements include other factors related to cost, weight, size, and power.

The error budget is an allocation of accuracy requirements down through the hierarchy of subsystems to individual sensors, and even to their component parts. It is used for a number of purposes, such as

1. Assessing theoretical or technological performance limits by determining whether the performance requirements for a given application of a given system are *achievable* within the performance capabilities of available, planned, or theoretically attainable sensor subsystems.
2. Determining the extent of *feasible design space*, which is the range of possible sensor types and their design parameters (e.g., placement, orientation, sensitivity, and accuracy) for meeting the system performance requirements.
3. Finding a *feasible apportionment* of individual subsystem or sensor accuracies for meeting overall system accuracy requirements.
4. Identifying the *critical* subsystems, that is, those for which slight degradation or derating of performance would most severely affect system performance. These are sometimes called *the long poles in the tent*, because they tend to "stick out" in this type of assessment.
5. Finding feasible *upgrades and redesigns* of existing systems for meeting new performance requirements. This may include relaxation of some requirements and tightening of others.
6. *Trading off* requirements among subsystems. This is done for a number of reasons:
   (a) Reapportionment of error budgets to meet a new set of requirements (item 5, above).
   (b) Relaxing accuracy requirements where they are difficult (or expensive) to attain and compensating by tightening requirements where they are easier to attain. This approach can sometimes be used to overcome sensor problems uncovered in concurrent development and testing.
   (c) Reducing other system-level performance attributes, such as cost, size, weight, and power. This also includes such practices as suboptimal filtering methods to reduce computational requirements.

9.9.2.1 Error Budget

Multiple Performance Requirements System-level performance requirements can include constraints on the mean-squared values of several error types at several different times. For example, the navigational errors of a space-based imaging system may be constrained at several points corresponding to photographic missions or planetary encounters. These constraints may include errors in pointing (attitude), position, and velocity. The error budget must then consider how each component, component group, or subsystem contributes to each of these performance requirements. The budget will then have a two-dimensional breakout—like a spreadsheet—as shown in Figure 9.30. The rows represent the contributions of major sensor subsystems, and the columns represent their contributions to each of the multiple system-level error

Error budget					
Error source group	System errors				
	E_1	E_2	E_3	...	E_n
G_1				...	
G_2				...	
G_3				...	
⋮	⋮	⋮	⋮	⋮	⋮
G_m				...	
Total				...	

Figure 9.30 Error budget breakdown.

constraints. The formulas determining how each error source contributes to each of the system-level error categories are more complex than those of the usual spreadsheet, however.

9.9.3 Error Sensitivity Analysis and Budgeting

A Nonlinear Programming Problem The dependence of mean-squared system-level errors on mean-squared subsystem-level errors is nonlinear, and the budgeting process seeks a satisfactory apportionment or the subsystem-level error covariances by a gradient-like method. This includes sensitivity analysis to determine the gradients of the various mean-squared system-level errors with respect to the mean-squared subsystem-level errors.

9.9.3.1 Dual-state System Modeling Errors considered in the error budgeting process may include known "modeling errors" due to simplifying assumptions or other measures to reduce the computational burden of the filter. For determining the effects that errors of this type will have on system performance, it is necessary to carry both models in the analysis: the "truth model" and the "filter model." The budgeting model used in this analysis is diagrammed in Figure 9.31. In sensitivity analysis, equivalent variations of some parameters must be made in both models. The resulting variations in the projected performance characteristics of the system are then used to establish the sensitivities to the corresponding variations in the subsystems. These sensitivities are then used to plan how one can modify the current "protobudget" to arrive at an error budget allocation that will meet all performance requirements. Often, this operation must be repeated many times, because the sensitivities estimated from variations are only accurate for small changes in the budget entries.

Two Stages of the Budgeting Process The first stage results in a "sufficing" error budget. It should meet system-level performance requirements and be reasonably close to attainable subsystem-level performance capabilities. The second stage includes

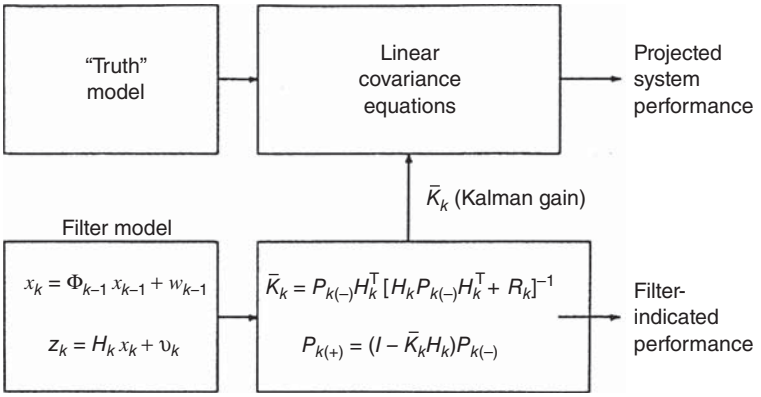

Figure 9.31 Error budgeting model.

"finessing" these subsystem-level error allocations to arrive at a more reasonable distribution.

9.9.4 Budget Validation by Monte Carlo Analysis

It is possible to validate some of the assumptions used in the error budgeting process by analytical and empirical methods. Although covariance analysis is more efficient for developing the error budget, Monte Carlo analysis is useful for assessing the effects of nonlinearities that have been approximated by variational models. This is typically done after the error budget is deemed satisfactory by linear methods. Monte Carlo analysis can then be performed on a dispersion of actual trajectories about some nominal trajectory to test the validity of the results estimated from the nominal trajectory. This is the only way to test the influence of nonlinearities, but it can be computationally expensive. Typically, very many Monte Carlo runs must be made to obtain reasonable confidence in the results.

Monte Carlo analysis has certain advantages over covariance analysis, however. The Monte Carlo simulations can be integrated with actual hardware, for example, to test the system performance in various stages of development. This is especially useful for testing filter performance in onboard computer implementations using actual system hardware as it becomes available. Sign errors in the filter algorithms that may be unimportant in covariance analysis will tend to show up under these test conditions.

9.10 OPTIMIZING MEASUREMENT SELECTION POLICIES

9.10.1 Measurement Selection Problem

9.10.1.1 Relation to Kalman Filtering and Error Budgeting We have seen how Kalman filtering solves the optimization problem related to the *use* of data obtained from a measurement and how error budgeting is used to quantify the relative merits of alternative sensor designs. However, there is an even more fundamental optimization

problem related to the *selection* of those measurements. This is not an estimation problem, strictly speaking, but a *decision problem*. It is usually considered to be a problem in the general theory of optimal control, because the decision to make a measurement is considered to be a generalized control action. The problem can also be ill-posed, in the sense that there may be no unique optimal solution [7].

9.10.1.2 Optimization with Respect to Quadratic Loss Function The Kalman filter is optimal with respect to all quadratic loss functions defining performance as a function of estimation error, but the measurement selection problem does not have that property. It depends very much on the particular loss function defining performance.

We present here a solution method based on what is called *maximum marginal benefit*. It is computationally efficient but suboptimal with respect to a given quadratic loss function of the resulting estimation errors $\hat{x} - x$:

$$\mathcal{L} = \sum_{\ell=1}^{N} \|A_\ell(\hat{x}_{\ell(+)} - x_\ell)\|^2, \tag{9.176}$$

where the given matrices A_ℓ transform the estimation errors to other "variables of interest," as illustrated by the following examples:

1. If only the *final values* of the estimation errors are of interest, then $A_N = I$ (the identity matrix) and $A_\ell = 0$ (a matrix of zeros) for $\ell < N$.
2. If only a *subset of the state vector components* are of interest, then the A_ℓ will all equal the projection onto those components that are of interest.
3. If *any linear transformation* of the estimation errors is of interest, then the A_ℓ will be defined by that transformation.
4. If any *temporally weighted combination* of linear transformations of the estimation errors is of interest, then the corresponding A_ℓ will be the weighted matrices of those linear transformation. That is, $A_\ell = f_\ell B_\ell$, where $0 \leq f_\ell$ is the temporal weighting and the B_ℓ are the matrices of the linear transformations.

9.10.2 Marginal Optimization

The loss function is defined above as a function of the a posteriori estimation errors following measurements. The next problem will be to represent the dependence of the associated risk[9] function on the selection of measurements.

9.10.2.1 Parameterizing the Possible Measurements As far as the Kalman filter is concerned, a measurement is characterized by H (its measurement sensitivity matrix)

[9]The term *risk* is here used to mean the expected loss.

and R (its covariance matrix of measurement uncertainty). A sequence of measurements is then characterized by the sequence

$$\{\{H_1, R_1\}, \{H_2, R_2\}, \{H_3, R_3\}, \ldots, \{H_N, R_N\}\}$$

of pairs of these parameters. This sequence will be called *marginally optimal* with respect to the above risk function if, for each k, the kth measurement is chosen to minimize the risk of the subsequence

$$\{\{H_1, R_1\}, \{H_2, R_2\}, \{H_3, R_3\}, \ldots, \{H_k, R_k\}\}.$$

That is, marginal optimization assumes that

1. The previous selections of measurements have already been decided.
2. No further measurements will be made after the current one is selected.

Admittedly, a marginally optimal solution is not necessarily a globally optimal solution. However, it does yield an efficient suboptimal solution method.

9.10.2.2 Marginal Risk Risk is the expected value of loss. The *marginal risk function* represents the functional dependence of risk on the selection of the kth measurement, *assuming that it is the last*. Marginal risk will depend only on the a posteriori estimation errors after the decision has been made. It can be expressed as an implicit function of the decision in the form

$$\mathcal{R}_k(P_k(+)) = E\left\{\sum_{\ell=k}^{N} \|A_\ell(\hat{x}_{\ell(+)} - x_\ell)\|^2\right\}, \tag{9.177}$$

where $P_k(+)$ will depend on the choice for the kth measurement and, for $k < \ell \leq N$,

$$\hat{x}_{\ell+1(+)} = \hat{x}_{\ell+1(-)}, \tag{9.178}$$

$$\hat{x}_{\ell+1(+)} - x_{\ell+1} = \Phi_\ell(\hat{x}_{\ell(+)} - x_\ell) - w_\ell, \tag{9.179}$$

so long as no additional measurements are used.

9.10.2.3 Marginal Risk Function Before proceeding further with the development of a solution method, it will be necessary to derive an explicit representation of the marginal risk as a function of the measurement used. For that purpose, one can use a *trace formulation* of the risk function, as presented in the following lemma.

Lemma 9.1 For $0 \leq k \leq N$, the risk function defined by Equation 9.177 can be represented in the form

$$\mathcal{R}_k(P_k) = \text{trace}\,\{P_k W_k + V_k\}, \tag{9.180}$$

where

$$W_N = A_N^{\mathrm{T}} A_N, \tag{9.181}$$

$$V_N = 0, \tag{9.182}$$

and, for $\ell < N$,

$$W_\ell = \Phi_\ell^{\mathrm{T}} W_{\ell+1} \Phi_\ell + A_\ell^{\mathrm{T}} A_\ell, \tag{9.183}$$

$$V_\ell = Q_\ell W_{\ell+1} + V_{\ell+1}. \tag{9.184}$$

Proof A formal proof of the equivalence of the two equations requires that each be entailed by (derivable from) the other. We give a proof here as a reversible chain of equalities, starting with one form and ending with the other form. This proof is by backward induction, starting with $k = N$ and proceeding by induction back to any $k \leq N$. The property that the trace of a matrix product is invariant under cyclical permutations of the order of multiplication is used extensively.

Initial step: The initial step of a proof by induction requires that the statement of the lemma hold for $k = N$. By substituting from Equations 9.181 and 9.182 into Equation 9.180, and substituting N for k, one can obtain the following sequence of equalities:

$$\begin{aligned}
\mathcal{R}_N(P_N) &= \mathrm{trace}\{P_N W_N + V_N\} \\
&= \mathrm{trace}\{P_N A_N^{\mathrm{T}} A_N + 0_{n\times n}\} \\
&= \mathrm{trace}\{A_N P_N A_N^{\mathrm{T}}\} \\
&= \mathrm{trace}\{A_N E\langle(\hat{x}_N - x_N)(\hat{x}_N - x_N)^{\mathrm{T}}\rangle A_N^{\mathrm{T}}\} \\
&= \mathrm{trace}\{\mathrm{E}\langle A_N(\hat{x}_N - x_N)(\hat{x}_N - x_N)^{\mathrm{T}} A_N^{\mathrm{T}}\rangle\} \\
&= \mathrm{trace}\{\mathrm{E}\langle[A_N(\hat{x}_N - x_N)][A_N(\hat{x}_N - x_N)]^{\mathrm{T}}\rangle\} \\
&= \mathrm{trace}\{\mathrm{E}\langle[A_N(\hat{x}_N - x_N)]^{\mathrm{T}}[A_N(\hat{x}_N - x_N)]\rangle\} \\
&= \mathrm{E}\langle\|A_N(\hat{x}_N - x_N)\|^2\rangle.
\end{aligned}$$

The first of these is Equation 9.180 for $k = N$, and the last is Equation 9.177 for $k = N$. That is, the statement of the lemma is true for $k = N$. This completes the initial step of the induction proof.

Induction step: One can suppose that Equation 9.180 is equivalent to Equation 9.177 for $k = \ell + 1$ and seek to prove from that it must also be the case for $k = \ell$. Then start with Equation 9.177, noting that it can be written in the form

$$\begin{aligned}
\mathcal{R}_\ell(P_\ell) &= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \mathrm{E}\langle\|A_\ell(\hat{x}_\ell - x_\ell)\|^2\rangle \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \mathrm{trace}\{\mathrm{E}\langle\|A_\ell(\hat{x}_\ell - x_\ell)\|^2\rangle\}
\end{aligned}$$

$$
\begin{aligned}
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{\text{E}\langle[A_\ell(\hat{x}_\ell - x_\ell)]^{\text{T}}[A_\ell(\hat{x}_\ell - x_\ell)]\rangle\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{\text{E}\langle[A_\ell(\hat{x}_\ell - x_\ell)][A_\ell(\hat{x}_\ell - x_\ell)]^{\text{T}}\rangle\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{A_\ell \text{E}\langle(\hat{x}_\ell - x_\ell)(\hat{x}_\ell - x_\ell)^{\text{T}}\rangle A_\ell^{\text{T}}\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{A_\ell P_\ell A_\ell^{\text{T}}\} \\
&= \mathcal{R}_{\ell+1}(P_{\ell+1}) + \text{trace}\{P_\ell A_\ell^{\text{T}} A_\ell\}.
\end{aligned}
$$

Now one can use the assumption that Equation 9.180 is true for $k = \ell + 1$ and substitute the resulting value for $\mathcal{R}_{\ell+1}$ into the last equation above. The result will be the following chain of equalities:

$$
\begin{aligned}
\mathcal{R}_\ell(P_\ell) &= \text{trace}\{P_{\ell+1}\ W_{\ell+1} + V_{\ell+1}\} + \text{trace}\{P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{P_{\ell+1}\ W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{[\Phi_\ell P_\ell \Phi_\ell^{\text{T}} + Q_\ell] W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{\Phi_\ell P_\ell \Phi_\ell^{\text{T}} W_{\ell+1} + Q_\ell W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{P_\ell \Phi_\ell^{\text{T}} W_{\ell+1} \Phi_\ell + Q_\ell W_{\ell+1} + V_{\ell+1} + P_\ell A_\ell^{\text{T}} A_\ell\} \\
&= \text{trace}\{P_\ell [\Phi_\ell^{\text{T}} W_{\ell+1} \Phi_\ell + A_\ell^{\text{T}} A_\ell] + [Q_\ell W_{\ell+1} + V_{\ell+1}]\} \\
&= \text{trace}\{P_\ell [W_\ell] + [V_\ell]\},
\end{aligned}
$$

where the Equations 9.183 and 9.184 were used in the last substitution. The last equation is Equation 9.180 with $k = \ell$, which was to be proved for the induction step. Therefore, by induction, the equations defining the marginal risk function are equivalent for $k \leq N$, which was to be proved.

Implementation Note The last formula separates the marginal risk as the sum of two parts. The first part depends only upon the choice of the measurement and the deterministic state dynamics. The second part depends only upon the stochastic state dynamics and is unaffected by the choice of measurements. As a consequence of this separation, the decision process will use only the first part. However, an assessment of the marginal risk performance of the decision process itself would require the evaluation of the complete marginal risk function.

9.10.2.4 Marginal Benefit The *marginal benefit* resulting from the use of a measurement will be defined as the associated *decrease* in the marginal risk. By this definition, the marginal benefit resulting from using a measurement with sensitivity matrix H and measurement uncertainty covariance R at time t_k will be the difference

between the a priori and a posteriori marginal risks:

$$\mathcal{B}(H,R) = \mathcal{R}_k(P_{k(-)}) - \mathcal{R}_k(P_{k(+)}) \tag{9.185}$$

$$= \text{trace}\{[P_{k(-)} - P_{k(+)}]W_k\} \tag{9.186}$$

$$= \text{trace}\{[P_{k(-)}H^{\mathrm{T}}(HP_{k(-)}H^{\mathrm{T}} + R)^{-1}HP_{k(-)}]W_k\} \tag{9.187}$$

$$= \text{trace}\{(HP_{k(-)}H^{\mathrm{T}} + R)^{-1}HP_{k(-)}W_kP_{k(-)}H^{\mathrm{T}}\}. \tag{9.188}$$

This last formula is in a form useful for implementation.

9.10.3 Solution Algorithm for Maximum Marginal Benefit

1. Compute the matrices W_ℓ using the formulas given by Equations 9.181 and 9.183.
2. Select the measurements in temporal order: for $k = 0, 1, 2, 3, \ldots, N$:
   (a) For each possible measurement, using Equation 9.188, evaluate the marginal benefit that would result from the use of that measurement.
   (b) Select the measurement that yields the *maximum* marginal benefit.

Again, note that this algorithm does *not* use the matrices V_ℓ in the "trace formulation" of the risk function. It is necessary to compute the V_ℓ only if the specific value of the associated risk is of sufficient interest to warrant the added computational expense.

9.10.4 Computational Complexity

9.10.4.1 Complexity of Computing the W_ℓ Complexity will depend upon the dimensions of the matrices A_ℓ. If each matrix A_ℓ is $p \times n$, then the products $A_\ell^{\mathrm{T}}A_\ell$ require $\mathcal{O}(pn^2)$ operations. The complexity of computing $\mathcal{O}(N)$ of the W_ℓ will then be $\mathcal{O}(Nn^2(p+n))$.

9.10.4.2 Complexity of Measurement Selection The computational complexity of making a single determination of the marginal benefit of a measurement of dimension m is summarized in Table 9.9. On each line, the complexity figure is based on reuse of partial results from computations listed on lines above. If all possible measurements have the same dimension ℓ and the number of such measurements to be evaluated is μ, then the complexity of evaluating all of them[10] will be $\mathcal{O}(\mu\ell(\ell^2 + n^2))$. If this is repeated for each of $\mathcal{O}(N)$ measurement selections, then the total complexity will be $\mathcal{O}(N\mu\ell(\ell^2 + n^2))$.

[10]Although the intermediate product $P_{k(-)}W_kP_{k(-)}$ (of complexity $\mathcal{O}(n^3)$) does not depend on the choice of the measurement, no reduction in complexity would be realized even if it were computed only once and reused for all measurements.

TABLE 9.9 Complexity of Determining the Marginal Benefit of a Measurement

Operation	Complexity
$HP_{k(-)}$	$\mathcal{O}(\ell n^2)$
$HP_{k(-)}H^{\mathrm{T}} + R$	$\mathcal{O}(\ell^2 n)$
$[HP_{k(-)}H^{\mathrm{T}} + R]^{-1}$	$\mathcal{O}(\ell^3)$
$HP_{k(-)}W_k$	$\mathcal{O}(\ell n^2)$
$HP_{k(-)}W_kP_{k(-)}H^{\mathrm{T}}$	$\mathcal{O}(\ell^2 n)$
$\mathrm{trace}\{(HP_{k(-)}H^{\mathrm{T}} + R)^{-1}HP_{k(-)}W_kP_{k(-)}H^{\mathrm{T}}\}$	$\mathcal{O}(\ell^2)$
Total	$\mathcal{O}(\ell(\ell^2 + n^2))$

Note: ℓ is the dimension of the measurement vector; n is the dimension of the state vector.

9.11 SUMMARY

This chapter discussed methods for the design and evaluation of estimation systems using Kalman filters. Specific topics addressed include the following:

1. methods for detecting and correcting anomalous behavior of estimators,
2. predicting and detecting the effects of mismodeling and poor unobservability,
3. evaluation of suboptimal filters (using dual-state filters) and sensitivity analysis methods,
4. comparison of memory, throughput, and wordlength requirements for alternative implementation methods,
5. methods for decreasing computational requirements,
6. methods for assessing the influence on estimator performance of sensor location and type and the number of sensors,
7. methods for top-down hierarchical system-level error budgeting, and
8. demonstration of the application of square-root filtering techniques to an inertial navigation system (INS)-aided GPS navigator.

PROBLEMS

9.1 Show that the final value of the risk obtained by the marginal optimization technique of Section 9.10 will equal the initial risk minus the sum of the marginal benefits of the measurements selected.

9.2 Develop the equations for the dual-state error propagation by substituting Equations 9.107 and 9.108 into Equations 9.111 and 9.112 using Equation 9.114, explicitly.

9.3 Obtain the dual-state vector equation for the covariances of the system and error, where x_1 is a ramp plus random walk and x_2 is constant:

$$\dot{x}_1^S = x_2^S + w^S, \quad \dot{x}_2^S = 0, \quad z_k = x_k^1 + v_k,$$

using as the filter model a random walk

$$\dot{x}^F = w^F, \quad z_K = x_k^F + v_k.$$

9.4 Derive the results of Example 7.4.

9.5 Prove that $\text{cov}[\tilde{x}_k^s]$ depends upon $\text{cov}(x_k^s)$.

9.6 Rework Problem 5.7 for the UDU^{T} formulation and compare your results with those of Problem 5.7.

9.7 Rework Problem 5.8 for the UDU^{T} formulation and compare your results with those of Problem 5.8.

9.8 Solve Problem 5.7 using the Schmidt–Kalman filter (Section 9.6) and compare the results with Example 5.8.

REFERENCES

[1] T. Kailath, "A general likelihood-ratio formula for random signals in Gaussian noise," *IEEE Transactions on Information Theory*, Vol. 15, No. 3, pp. 350–361, 1969.

[2] M. S. Grewal, V. D. Henderson, and R. S. Miyasako, "Application of Kalman filtering to the calibration and alignment of inertial navigation systems," *IEEE Transactions on Automatic Control*, Vol. AC-38, pp. 4–13, 1991.

[3] A. Gelb, J. F. Kasper Jr., R. A. Nash Jr., C. F. Price, and A. A. Sutherland Jr., *Applied Optimal Estimation*, MIT Press, Cambridge, MA, 1974.

[4] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, Vol. 1, Academic Press, New York, 1979.

[5] S. F. Schmidt, "Applications of state-space methods to navigation problems," in *Advances in Control Systems*, Vol. 3 (C. T. Leondes, ed.), Academic Press, New York, pp. 293–340, 1966.

[6] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 4th ed., John Wiley & Sons, Inc., New York, 2012.

[7] A. Andrews, "Marginal optimization of observation schedules," *AIAA Journal of Guidance and Control*, Vol. 5, pp. 95–96, 1982.