Chapter 4

Polynomial Kalman Filters

Introduction

IN THE previous chapter we saw how the batch-processing method of least-squares could be made recursive when the measurement was a polynomial signal corrupted by noise. We saw that the recursive least-squares filter provided the exact same estimates, for a given number of measurements, as the batch-processing least-squares filter. In this chapter we will first provide the general equations for the discrete Kalman filter and then show under what conditions it is completely equivalent to the recursive least-squares filter.

We also saw in the preceding chapter that if the order of the recursive least-squares filter was less than the order of the polynomial measurement signal then the filter's estimates would diverge from the true signal and its derivatives. The engineering fix was to use a higher-order filter. We will also demonstrate the divergence problem with the Kalman filter, but we will show that there are a variety of engineering fixes.

General Equations[1,2]

To apply Kalman-filtering theory, our model of the real world must be described by a set of differential equations. These equations must be cast in matrix or state-space form as

$$\dot{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}\boldsymbol{u} + \boldsymbol{w}$$

where $\boldsymbol{x}$ is a column vector with the states of the system, $\boldsymbol{F}$ the system dynamics matrix, $\boldsymbol{u}$ is a known vector, which is sometimes called the control vector, and $\boldsymbol{w}$ is a white-noise process, which is also expressed as a vector. There is a process-noise matrix $\boldsymbol{Q}$ that is related to the process-noise vector according to

$$\boldsymbol{Q} = \boldsymbol{E}[\boldsymbol{w}\boldsymbol{w}^T]$$

We will also see that, although process noise might not always have physical meaning, it is sometimes used as a device for telling the filter that we know the filter's model of the real world is not precise. The Kalman-filter formulation requires that the measurements be linearly related to the states according to

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{v}$$

where **z** is the measurement vector, $\boldsymbol{H}$ is the measurement matrix, and $\boldsymbol{v}$ is white measurement noise, which is also expressed as a vector. The measurement noise matrix $\boldsymbol{R}$ is related to the measurement noise vector $\boldsymbol{v}$ according to

$$\boldsymbol{R} = \boldsymbol{E}[\boldsymbol{v}\boldsymbol{v}^T]$$

The preceding relationships must be discretized before a discrete Kalman filter can be built. If we take measurements every T_s seconds, we first need to find a fundamental matrix Φ. Recall from Chapter 1 that for a time-invariant system the fundamental matrix can be found from the system dynamics matrix according to

$$\Phi(\boldsymbol{t}) = \mathscr{L}^{-1}[(s\boldsymbol{I} - \boldsymbol{F})^{-1}]$$

where $\boldsymbol{I}$ is the identity matrix, $\mathscr{L}^{-1}$ is the inverse Laplace transform, and $\boldsymbol{F}$ is the systems dynamics matrix. Typically, inverse Laplace transforms can be found from tables in engineering handbooks.[3] As was also mentioned in Chapter 1, another way of finding the fundamental matrix is to evaluate the Taylor-series expansion

$$\Phi(t) = e^{\boldsymbol{F}t} = \boldsymbol{I} + \boldsymbol{F}t + \frac{(\boldsymbol{F}t)^2}{2!} + \cdots + \frac{(\boldsymbol{F}t)^n}{n!} + \cdots$$

The discrete fundamental or transition matrix can easily be found by evaluating the fundamental matrix at the sampling time T_s or

$$\Phi_k = \Phi(T_s)$$

The discrete form of the Kalman-filtering measurement equation becomes

$$\boldsymbol{z}_k = \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k$$

and

$$\boldsymbol{R}_k = E(\boldsymbol{v}_k\boldsymbol{v}_k^T)$$

where $\boldsymbol{R}_k$ is a matrix consisting of the variances of each of the measurement noise sources. In the case of polynomial Kalman filters, $\boldsymbol{R}_k$ is a scalar. The resultant Kalman-filtering equation is given by

$$\hat{\boldsymbol{x}}_k = \Phi_k\hat{\boldsymbol{x}}_{k-1} + \boldsymbol{G}_k\boldsymbol{u}_{k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\Phi_k\hat{\boldsymbol{x}}_{k-1} - \boldsymbol{H}\boldsymbol{G}_k\boldsymbol{u}_{k-1})$$

where $\boldsymbol{K}_k$ represents the Kalman gain matrix and $\boldsymbol{G}_k$ is obtained from

$$\boldsymbol{G}_k = \int_0^{T_s} \Phi(\tau)\boldsymbol{G}\,\mathrm{d}\tau$$

if $\boldsymbol{u}_{k-1}$ is assumed to be constant between sampling instants. The Kalman gains are computed, while the filter is operating, from the matrix Riccati equations. The Riccati equations are a set of recursive matrix equations given by

$$\boldsymbol{M}_k = \Phi_k \boldsymbol{P}_{k-1} \Phi_k^T + \boldsymbol{Q}_k$$
$$\boldsymbol{K}_k = \boldsymbol{M}_k \boldsymbol{H}^T (\boldsymbol{H}\boldsymbol{M}_k \boldsymbol{H}^T + \boldsymbol{R}_k)^{-1}$$
$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})\boldsymbol{M}_k$$

where $\boldsymbol{P}_k$ is a covariance matrix representing errors in the state estimates (i.e., variance of truth minus estimate) after an update and $\boldsymbol{M}_k$ is the covariance matrix representing errors in the state estimates before an update. The discrete process-noise matrix $\boldsymbol{Q}_k$ can be found from the continuous process-noise matrix $\boldsymbol{Q}$ and the fundamental matrix according to

$$\boldsymbol{Q}_k = \int_0^{T_s} \Phi(\tau) \boldsymbol{Q} \Phi^T(\tau)\, \mathrm{d}t$$

To start the Riccati equations, we need an initial covariance matrix $\boldsymbol{P}_0$.

Derivation of Scalar Riccati Equations

A special form of the Kalman filter occurs when there is no deterministic disturbance or control vector. In this special case the discrete Kalman-filtering equation simplifies to

$$\hat{\boldsymbol{x}}_k = \Phi_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\Phi \hat{\boldsymbol{x}}_{k-1})$$

while the matrix Riccati equations remain as

$$\boldsymbol{M}_k = \Phi_k \boldsymbol{P}_{k-1} \Phi_k^T + \boldsymbol{Q}_k$$
$$\boldsymbol{K}_k = \boldsymbol{M}_k \boldsymbol{H}^T (\boldsymbol{H}\boldsymbol{M}_k \boldsymbol{H}^T + \boldsymbol{R}_k)^{-1}$$
$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H})\boldsymbol{M}_k$$

Although the derivation of the matrix Riccati equations is beyond the scope of this text, the derivation process to illustrate when the matrices are scalars is quite simple. If there is no deterministic disturbance or control scalar, our discrete model of the real world is given by

$$x_k = \Phi_k x_{k-1} + w_k$$

where Φ_k is the scalar that propagates the states from one sampling instant to the next and w_k is white process noise. From the first equation of this section, we can see the scalar Kalman-filtering equation is

$$\hat{x}_k = \Phi_k \hat{x}_{k-1} + K_k(z_k - H\Phi_k \hat{x}_{k-1})$$

Our zeroth-order recursive least-squares filter of Chapter 3 was of the same form. In the preceding scalar equation z_k is the measurement, which is given by

$$z_k = Hx_k + v_k$$

where H is the measurement scalar that relates the state to the measurement and v_k is the measurement noise. From the preceding equations we can say that the error in the estimate is

$$\tilde{x}_k = x_k - \hat{x}_k = x_k - \Phi_k \hat{x}_{k-1} - K_k(z_k - H\Phi_k \hat{x}_{k-1})$$

Recognizing that we can express the measurement in terms of the state, we obtain

$$\tilde{x}_k = x_k - \Phi_k \hat{x}_{k-1} - K_k(Hx_k + v_k - H\Phi_k \hat{x}_{k-1})$$

If we also note that the state at time k can be replaced by an alternate expression at time $k-1$, we get

$$\tilde{x}_k = \Phi_k x_{k-1} + w_k - \Phi_k \hat{x}_{k-1} - K_k(H\Phi_k x_{k-1} + Hw_k + v_k - H\Phi_k \hat{x}_{k-1})$$

Because

$$\tilde{x}_k = x_k - \hat{x}_k$$

we can also say that

$$\tilde{x}_{k-1} = x_{k-1} - \hat{x}_{k-1}$$

Therefore, combining similar terms in the error in the estimate equation yields

$$\tilde{x}_k = (1 - K_k H)\tilde{x}_{k-1}\Phi_k + (1 - K_k H)w_k - K_k v_k$$

If we define the covariance P_k to be

$$P_k = E(\tilde{x}_k^2)$$

and recognize that

$$Q_k = E(w_k^2)$$
$$R_k = E(v_k^2)$$

we can square and take expectations of both sides of the error in the estimate equation to obtain

$$P_k = (1 - K_k H)^2 (P_{k-1}\Phi_k^2 + Q_k) + K_k^2 R_k$$

To simplify the preceding equation, let us define

$$M_k = P_{k-1}\Phi_k^2 + Q_k$$

which is analogous to the first Riccati equation. The covariance equation now simplifies to

$$P_k = (1 - K_k H)^2 M_k + K_k^2 R_k$$

If we want to find the gain that will minimize the variance of the error in the estimate, we can simply take the derivative of the preceding expression with respect to the gain and set the result equal to zero or

$$\frac{\partial P_k}{\partial K_k} = 0 = 2(1 - K_k H)M_k(-H) + 2K_k R_k$$

Solving the preceding equation for the gain yields

$$K_k = \frac{M_k H}{H^2 M_k + R_k} = M_k H(H^2 M_k + R_k)^{-1}$$

which is analogous to the second Riccati equation. Substitution of the optimal gain into the covariance equation yields

$$P_k = \left(1 - \frac{M_k H^2}{H^2 M_k + R_k}\right)M_k + \left(\frac{M_k H}{H^2 M_k + R_k}\right)^2 R_k$$

which simplifies to

$$P_k = \frac{R_k M_k}{H^2 M_k + R_k} = \frac{R_k K_k}{H}$$

By inverting the optimal gain equation, we obtain

$$K_k R_k = M_k H - H^2 M_k K_k$$

and substituting the preceding equation back into the variance equation yields

$$P_k = \frac{R_k K_k}{H} = \frac{M_k H - H^2 M_k K_k}{H} = M_k - H M_k K_k$$

or, more simply,

$$P_k = (1 - K_k H)M_k$$

which is analogous to the third Riccati equation.

From the preceding derivation we can see that the gain of the Kalman filter is chosen to minimize the variance of the error in the estimate. The Riccati equations are simply an iterative way of finding the optimal gain at each time step. However, the derivation of the matrix Riccati equations is far more complex than the scalar derivation of this section because matrices do not always obey the same rules as scalars. For example, with matrices the order of their place in multiplication is important, whereas this is not true with scalars (i.e., $\boldsymbol{P}_1\boldsymbol{P}_2 \neq \boldsymbol{P}_2\boldsymbol{P}_1$, whereas $P_1P_2 = P_2P_1$). In addition, the matrix inverse cannot be computed in the same way in which we computed the scalar inverse.

Polynomial Kalman Filter (Zero Process Noise)

We have already mentioned that a special form of the Kalman filter occurs when there is no control or deterministic vector. If the differential equations describing system behavior yield polynomial signals, the resultant polynomial Kalman filter will have a special type of fundamental matrix. The simplest possible polynomial filter occurs when there is no process noise (i.e., $\boldsymbol{Q}_k = 0$). In this special case the filtering equation simplifies to

$$\hat{x}_k = \Phi_k\hat{x}_{k-1} + \boldsymbol{K}_k(z_k - \boldsymbol{H}\Phi_k\hat{\boldsymbol{x}}_{k-1})$$

while the Riccati equations simplify to

$$\begin{aligned} \boldsymbol{M}_k &= \Phi_k\boldsymbol{P}_{k-1}\Phi_k^T \\ \boldsymbol{K}_k &= \boldsymbol{M}_k\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{M}_k\boldsymbol{H}^T + \boldsymbol{R}_k)^{-1} \\ \boldsymbol{P}_k &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\boldsymbol{M}_k \end{aligned}$$

If the actual signal or polynomial is a constant, then we can say that

$$x = a_0$$

which means that the derivative of the signal is zero or

$$\dot{x} = 0$$

If the state-space representation of the preceding equation is given by

$$\dot{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{x}$$

then we can say for this zeroth-order system that

$$\boldsymbol{F} = 0$$

If the polynomial under consideration was a ramp or

$$x = a_0 + a_1t$$

then its derivative is given by

$$\dot{x} = a_1$$

whereas its second derivative is given by

$$\ddot{x} = 0$$

We can express the preceding two equations in state-space form as

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

which means the system dynamics matrix for a ramp or first-order polynomial is given by

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

We can proceed using similar logic assuming the signal is a parabola. That work has been done and Table 4.1 tabulates the system dynamics matrix, the resultant fundamental matrix, the measurement matrix, and the noise matrix (scalar) for different-order systems, where it is assumed that the measurement is a polynomial signal plus noise.

The matrices of Table 4.1 form the basis of the different-order polynomial Kalman filters. For the polynomial Kalman filters considered, the fundamental matrices can be derived exactly from the system dynamics matrix according to

$$\Phi(\boldsymbol{t}) = \mathscr{L}^{-1}[(s\boldsymbol{I} - \boldsymbol{F})^{-1}]$$

Table 4.1 Important matrices for different-order polynomial Kalman filters

Order	Systems dynamics	Fundamental	Measurement	Noise
0	$\boldsymbol{F} = 1$	$\Phi_k = 1$	$\boldsymbol{H} = 1$	$\boldsymbol{R}_k = \sigma_n^2$
1	$\boldsymbol{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$	$\Phi_k = \begin{bmatrix} 1 & \boldsymbol{T}_s \\ 0 & 1 \end{bmatrix}$	$\boldsymbol{H} = [1 \quad 0]$	$\boldsymbol{R}_k = \sigma_n^2$
2	$\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$\Phi_k = \begin{bmatrix} 1 & T_s & 0.5T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}$	$\boldsymbol{H} = [1 \quad 0 \quad 0]$	$\boldsymbol{R}_k = \sigma_n^2$

or exactly from the Taylor-series expansion

$$\Phi(t) = e^{\boldsymbol{F}t} = \boldsymbol{I} + \boldsymbol{F}t + \frac{(\boldsymbol{F}t)^2}{2!} + \cdots + \frac{(\boldsymbol{F}t)^n}{n!} + \cdots$$

For example, in the first-order or two-state system the systems dynamics matrix is given by

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

Therefore, we can find the square of the systems dynamics matrix to be

$$\boldsymbol{F}^2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

which means that the fundamental matrix as a function of time is exactly a three-term Taylor-series expansion or

$$\Phi(t) = e^{\boldsymbol{F}t} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} t + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \frac{t^2}{2} = \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix}$$

Substituting T_s for t in the preceding equation yields the discrete form of the fundamental matrix to be

$$\Phi_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$$

which is precisely what appears in Table 4.1. The $\boldsymbol{H}$ and $\boldsymbol{R}_k$ matrices shown in Table 4.1 indicate that for the polynomial Kalman filters considered the measurement is always the first state corrupted by noise with variance σ_n^2. Therefore $\boldsymbol{R}_k$ is a scalar, which means it will be easy to take the matrix inverse as required by the second Riccati equation. In this case the matrix inverse and scalar inverse are identical.

Comparing Zeroth-Order Recursive Least-Squares and Kalman Filters

Table 4.1 provides enough information so that we can examine the structure of the polynomial Kalman filter. First we will consider the zeroth-order filter. Substitution of the fundamental and measurement matrices of Table 4.1 into the Kalman-filtering equation

$$\hat{\boldsymbol{x}}_k = \Phi_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{K}_k(z_k - \boldsymbol{H}\Phi_k \hat{\boldsymbol{x}}_{k-1})$$

yields the scalar equation

$$\hat{x}_k = \hat{x}_{k-1} + K_{1_k}(x_k^* - \hat{x}_{k-1})$$

If we define the residual Res_k as

$$\text{Res}_k = x_k^* - \hat{x}_{k-1}$$

then the preceding zeroth-order or one-state polynomial Kalman-filter equation simplifies to

$$\hat{x}_k = \hat{x}_{k-1} + K_{1_k}\text{Res}_k$$

which is the same as the equation for the zeroth-order recursive least-squares filter of the preceding chapter. The only possible difference between the zeroth-order recursive least-squares filter and polynomial Kalman filter is in the gain.

Table 4.1 also provides enough information so that we can solve the Riccati equations for the Kalman gains. If we assume that the initial covariance matrix is infinite (i.e., this means that we have no a priori information on how to initialize the filter state), then for the zeroth-order system we have

$$P_0 = \infty$$

From the first Riccati equation we can say that

$$M_1 = \Phi_1 P_0 \Phi_1^T = 1{*}\infty{*}1 = \infty$$

Therefore, the second Riccati equation tells us that

$$K_1 = M_1 H^T (H M_1 H^T + R_1)^{-1} = \frac{M_1}{M_1 + \sigma_n^2} = \frac{\infty}{\infty + \sigma_n^2} = 1$$

The zeroth-order polynomial Kalman-filter gain is the same value of the gain with $k = 1$ as the zeroth-order recursive least-squares filter of Chapter 3. Solving the third Riccati equation yields

$$P_1 = (I - K_1 H) M_1 = \left(1 - \frac{M_1}{M_1 + \sigma_n^2}\right) M_1 = \frac{\sigma_n^2 M_1}{M_1 + \sigma_n^2} = \sigma_n^2$$

which is also the same value for the variance of the error in the estimate obtained for $k = 1$ with the zeroth-order recursive least-squares filter of Chapter 3. Repeating the first Riccati equation for $k = 2$ yields

$$M_2 = \Phi_2 P_1 \Phi_2^T = 1{*}\sigma_n^2{*}1 = \sigma_n^2$$

Substitution into the second Riccati equation yields the Kalman gain

$$K_2 = \frac{M_2}{M_2 + \sigma_n^2} = \frac{\sigma_n^2}{\sigma_n^2 + \sigma_n^2} = 0.5$$

Again, the second value of the zeroth-order polynomial Kalman-filter gain is the same value of the gain with $k = 2$ in the zeroth-order recursive least-squares filter of Chapter 3. Solving the third Riccati equation yields the covariance of the error in the estimate

$$P_2 = \frac{\sigma_n^2 M_2}{M_2 + \sigma_n^2} = \frac{\sigma_n^2 \sigma_n^2}{\sigma_n^2 + \sigma_n^2} = \frac{\sigma_n^2}{2}$$

which is also the same value for the variance of the error in the estimate obtained for $k = 2$ with the zeroth-order recursive least-squares filter of Chapter 3. Repeating the first Riccati equation for $k = 3$ yields

$$M_3 = P_2 = \frac{\sigma_n^2}{2}$$

Solving the second Riccati equation for the Kalman gain yields

$$K_3 = \frac{M_3}{M_3 + \sigma_n^2} = \frac{0.5\sigma_n^2}{0.5\sigma_n^2 + \sigma_n^2} = \frac{1}{3}$$

Again, this is the same value of the gain with $k = 3$ in the zeroth-order recursive least-squares filter. Solving the third Riccati equation

$$P_3 = \frac{\sigma_n^2 M_3}{M_3 + \sigma_n^2} = \frac{\sigma_n^2 * 0.5\sigma_n^2}{0.5\sigma_n^2 + \sigma_n^2} = \frac{\sigma_n^2}{3}$$

which is also the same value for the variance of the error in the estimate obtained for $k = 3$ with the zeroth-order recursive least-squares filter of Chapter 3. Now we solve the Riccati equations the last time for $k = 4$. Solving the first equation yields

$$M_4 = P_3 = \frac{\sigma_n^2}{3}$$

Solving the second equation yields the Kalman gain

$$K_4 = \frac{M_4}{M_4 + \sigma_n^2} = \frac{0.333\sigma_n^2}{0.333\sigma_n^2 + \sigma_n^2} = \frac{1}{4}$$

which again is the same as the zeroth-order recursive least-squares filter gain with $k = 4$. Solving the last Riccati equation yields

$$P_4 = \frac{\sigma_n^2 M_4}{M_4 + \sigma_n^2} = \frac{\sigma_n^2 * 0.333\sigma_n^2}{0.333\sigma_n^2 + \sigma_n^2} = \frac{\sigma_n^2}{4}$$

which is also the same value for the variance of the error in the estimate obtained for $k = 3$ with the zeroth-order recursive least-squares filter of the preceding chapter. *Thus, when the zeroth-order polynomial Kalman filter has zero process noise and infinite initial covariance matrix, it has the same gains and variance predictions as the zeroth-order recursive least-squares filter. We will soon see that this is not a coincidence.* Our particular recursive least-squares filter was developed without any assumption concerning the statistical information about the states being estimated.

Comparing First-Order Recursive Least-Squares and Kalman Filters

Table 4.1 also provides enough information so that we can examine the structure of the first-order polynomial Kalman filter. Substitution of the fundamental and measurement matrices of Table 4.1 into

$$\hat{\boldsymbol{x}}_k = \Phi_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{K}_k(z_k - \boldsymbol{H}\Phi_k \hat{\boldsymbol{x}}_{k-1})$$

yields

$$\begin{bmatrix} \hat{x}_k \\ \hat{\dot{x}}_k \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \end{bmatrix} + \begin{bmatrix} K_{1_k} \\ K_{2_k} \end{bmatrix} \left[x_k^* - [1 \quad 0] \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \end{bmatrix} \right]$$

Multiplying out the terms of the matrix equation yields the two scalar equations

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k-1} + T_s \hat{\dot{x}}_{k-1} + K_{1_k}(x_k^* - \hat{x}_{k-1} - T_s \hat{\dot{x}}_{k-1}) \\ \hat{\dot{x}}_k &= \hat{\dot{x}}_{k-1} + K_{2_k}(x_k^* - \hat{x}_{k-1} - T_s \hat{\dot{x}}_{k-1}) \end{aligned}$$

If we define the residual Res_k as

$$\text{Res}_k = x_k^* - \hat{x}_{k-1} - T_s \hat{\dot{x}}_{k-1}$$

then the two preceding equations simplify to

$$\begin{aligned} \hat{x}_k &= \hat{x}_{k-1} + T_s \hat{\dot{x}}_{k-1} + K_{1_k} \text{Res}_k \\ \hat{\dot{x}}_k &= \hat{\dot{x}}_{k-1} + K_{2_k} \text{Res}_k \end{aligned}$$

which is precisely the same as the equations for the first-order recursive least-squares filter of Chapter 3. Again, the only possible difference between the first-order recursive least-squares filter and polynomial Kalman filter are in the gains.

Next, we would like to compare the gains and variance predictions of both the first-order recursive least-squares filter and the first-order Kalman filter with zero process-noise and infinite initial covariance matrix. Recall that the formula presented in Chapter 3 for the first-order recursive least-square filter gains is given by

$$K_{1_k} = \frac{2(2k-1)}{k(k+1)} \qquad k = 1, 2, \ldots, n$$

$$K_{2_k} = \frac{6}{k(k+1)T_s}$$

whereas the formula for the variance of the errors in the estimates (i.e., variance of truth minus estimate) of the first and second states is given by

$$P_{11_k} = \frac{2(2k-1)\sigma_n^2}{k(k+1)}$$

$$P_{22_k} = \frac{12\sigma_n^2}{k(k^2-1)T_s^2}$$

A program was written in which the Riccati equations were solved for the first-order polynomial Kalman filter and is presented in Listing 4.1. The Riccati equations not only yield values for the Kalman gains but also values for the entire covariance matrix $\boldsymbol{P}_k$. Logic was added to the program so that the gains and square root of the covariance matrix diagonal elements computed from Listing 4.1 could be compared to the preceding set of formulas for the first-order recursive least-squares filter.

Figure 4.1 compares the square root of the first diagonal element of the covariance matrix of the first-order polynomial Kalman filter to the square root of the formula for variance of the error in the estimate of the first state in the first-order recursive least-squares filter. We can see that the comparison is exact, meaning that the filters are identical when the Kalman filter has zero process noise and infinite initial covariance matrix (i.e., infinite value for diagonal terms and zero value for off-diagonal terms). In addition, this plot also represents how the standard deviation of the error in the estimate of the first state (i.e., truth minus estimate) improves as more measurements are taken. In this example the input noise had a standard deviation of unity. After approximately 15 measurements the standard deviation of the error in the estimate decreases to approximately 0.5, meaning that filtering has effectively reduced the effect of noise by a factor of two.

Figure 4.2 shows that both the first-order polynomial Kalman and recursive least-squares filters also agree exactly in their predictions of the standard deviation in the error in the estimate of the second state (i.e., derivative of first state). We can also see that the standard deviation in the error of the estimate of the second state diminishes quite rapidly as more measurements are taken.

We can also see from Figures 4.3 and 4.4 that both gains of the first-order polynomial Kalman and recursive least-squares filters agree exactly. *In other words, we have confirmed by simulation that the first-order polynomial Kalman filter with zero process-noise and infinite initial covariance matrix and the*

Listing 4.1 Comparing first-order polynomial Kalman and recursive least-squares filter gains and covariance matrices

```
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 M(2,2),P(2,2),K(2,1),PHI(2,2),H(1,2),R(1,1),PHIT(2,2)
      REAL*8 PHIP(2,2),HT(2,1),KH(2,2),IKH(2,2)
      REAL*8 MHT(2,1),HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),IDN(2,2)
      REAL*8 K1GM,K2GM
      INTEGER ORDER
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      ORDER=2
      TS=1.
      SIGNOISE=1.
      DO 1000 I=1,ORDER
      DO 1000 J=1,ORDER
               PHI(I,J)=0.
               P(I,J)=0.
               IDN(I,J)=0.
1000  CONTINUE
      IDN(1,1)=1.
      IDN(2,2)=1.
      P(1,1)=99999999999999.
      P(2,2)=99999999999999.
      PHI(1,1)=1
      PHI(1,2)=TS
      PHI(2,2)=1
      DO 1100 I=1,ORDER
               H(1,I)=0.
 1100 CONTINUE
      H(1,1)=1
      CALL MATTRN(H,1,ORDER,HT)
      R(1,1)=SIGNOISE**2
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      DO 10 XN=1.,100.
            CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
            CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,M)
            CALL MATMUL(M,ORDER,ORDER,HT,ORDER,1,MHT)
            CALL MATMUL(H,1,ORDER,MHT,ORDER,1,HMHT)
            HMHTR(1,1)=HMHT(1,1)+R(1,1)
            HMHTRINV(1,1)=1./HMHTR(1,1)
            CALL MATMUL(MHT,ORDER,1,HMHTRINV,1,1,K)
            CALL MATMUL(K,ORDER,1,H,1,ORDER,KH)
            CALL MATSUB(IDN,ORDER,ORDER,KH,IKH)
            CALL MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
            IF(XN<2)THEN
                  P11GM=9999999999.
                  P22GM=9999999999.
            ELSE
                  P11GM=2.*(2.*XN-1)*SIGNOISE*SIGNOISE/(XN*(XN+1.))
                  P22GM=12.*SIGNOISE*SIGNOISE/(XN*(XN*XN-1.)
1                        *TS*TS)
            ENDIF
            SP11=SQRT(P(1,1))
```

(*continued*)

Listing 4.1 (*Continued*)

```
          SP22=SQRT(P(2,2))
          SP11GM=SQRT(P11GM)
          SP22GM=SQRT(P22GM)
          K1GM=2.*(2.*XN-1.)/(XN*(XN+1.))
          K2GM=6./(XN*(XN+1.)*TS)
          WRITE(9,*)XN,K(1,1),K1GM,K(2,1),K2GM,SP11,SP11GM,SP22,
1                   SP22GM
          WRITE(1,*)XN,K(1,1),K1GM,K(2,1),K2GM,SP11,SP11GM,SP22,
1                   SP22GM
10    CONTINUE
      CLOSE(1)
      PAUSE
      END

C SUBROUTINE MATTRN IS SHOWN IN LISTING 1.3
C SUBROUTINE MATMUL IS SHOWN IN LISTING 1.4
C SUBROUTINE MATADD IS SHOWN IN LISTING 1.1
C SUBROUTINE MATSUB IS SHOWN IN LISTING 1.2
```

recursive polynomial least-squares filters not only have identical structure but also have identical gains.

Comparing Second-Order Recursive Least-Squares and Kalman Filters

Table 4.1 also provides enough information so that we can examine the structure of the second-order polynomial Kalman filter. Substitution of the fundamental and measurement matrices of Table 4.1 into the fundamental Kalman-filtering equation

$$\hat{\boldsymbol{x}}_k = \Phi_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{K}_k(z_k - \boldsymbol{H}\Phi_k \hat{\boldsymbol{x}}_{k-1})$$

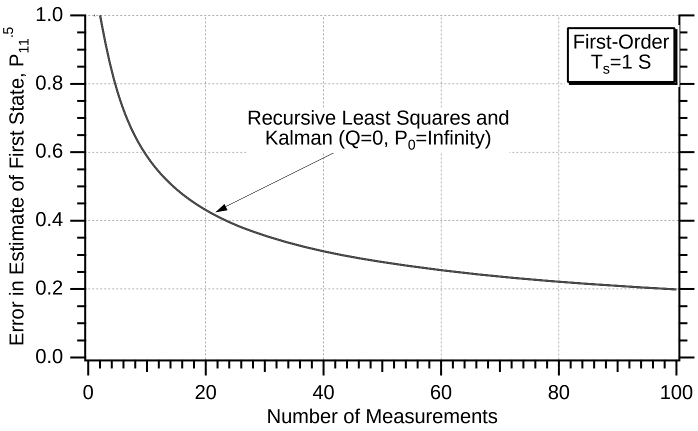

Fig. 4.1 First-order polynomial Kalman and recursive least-squares filters have identical standard deviations for errors in the estimate of the first state.

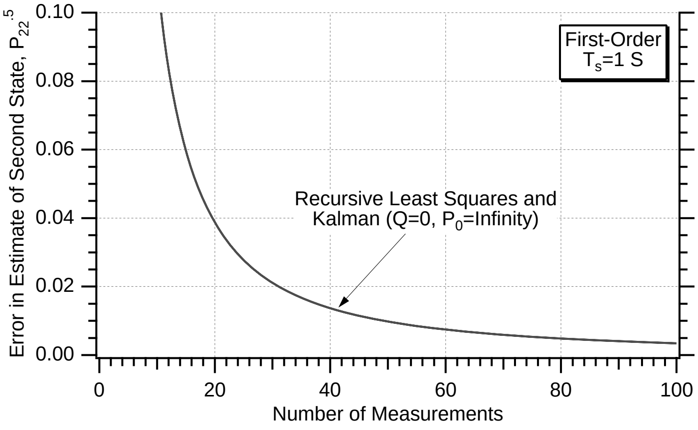

Fig. 4.2 First-order polynomial Kalman and recursive least-squares filters have identical standard deviations for errors in the estimate of the second state.

yields

$$\begin{bmatrix}\hat{x}_k\\ \hat{\dot{x}}_k\\ \hat{\ddot{x}}_k\end{bmatrix}=\begin{bmatrix}1 & T_s & 0.5T_s^2\\ 0 & 1 & T_s\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\hat{x}_{k-1}\\ \hat{\dot{x}}_{k-1}\\ \hat{\ddot{x}}_{k-1}\end{bmatrix}$$
$$+\begin{bmatrix}K_{1_k}\\ K_{2_k}\\ K_{3_k}\end{bmatrix}\left[x_k^*-[1\quad 0\quad 0]\begin{bmatrix}1 & T_s & 0.5T_s^2\\ 0 & 1 & T_s\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\hat{x}_{k-1}\\ \hat{\dot{x}}_{k-1}\\ \hat{\ddot{x}}_{k-1}\end{bmatrix}\right]$$

Multiplying out the terms of the preceding matrix equation yields the three scalar equations for the second-order or three-state polynomial Kalman filter

$$\hat{x}_k=\hat{x}_{k-1}+T_s\hat{\dot{x}}_{k-1}+0.5T_s^2\hat{\ddot{x}}_{k-1}+K_{1_k}(x_k^*-\hat{x}_{k-1}-T_s\hat{\dot{x}}_{k-1}-0.5T_s^2\hat{\ddot{x}}_{k-1})$$
$$\hat{\dot{x}}_k=\hat{\dot{x}}_{k-1}+T_s\ddot{x}_{k-1}+K_{2_k}(x_k^*-\hat{x}_{k-1}-T_s\hat{\dot{x}}_{k-1}-.5T_s^2\hat{\ddot{x}}_{k-1})$$
$$\hat{\ddot{x}}_k=\hat{\ddot{x}}_{k-1}+K_{3_k}(x_k^*-\hat{x}_{k-1}-T_s\hat{\dot{x}}_{k-1}-.5T_s^2\hat{\ddot{x}}_{k-1})$$

If we define the residual Res_k as

$$\mathrm{Res}_k=x_k^*-\hat{x}_{k-1}-T_s\hat{\dot{x}}_{k-1}-.5T_s^2\hat{\ddot{x}}_{k-1}$$

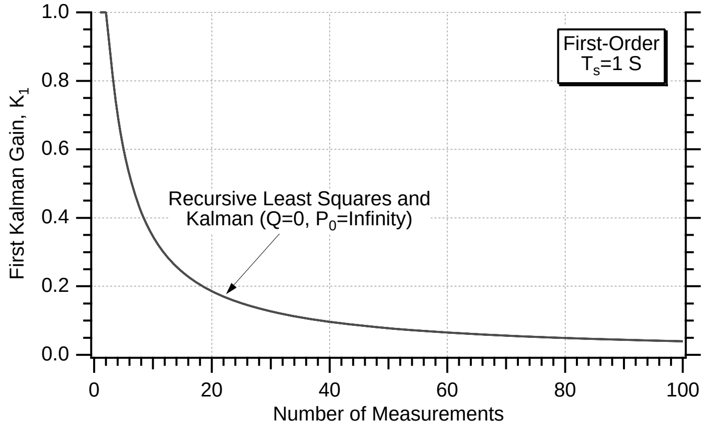

Fig. 4.3 First gain of first-order polynomial Kalman and recursive least-squares filters are identical.

then the three preceding equations simplify to

$$\hat{x}_k = \hat{x}_{k-1} + T_s\hat{\dot{x}}_{k-1} + 0.5T_s^2\hat{\ddot{x}}_{k-1} + K_{1_k}\mathrm{Res}_k$$
$$\hat{\dot{x}}_k = \hat{\dot{x}}_{k-1} + T_s\hat{\ddot{x}}_{k-1} + K_{2_k}\mathrm{Res}_k$$
$$\hat{\ddot{x}}_k = \hat{\ddot{x}}_{k-1} + K_{3_k}\mathrm{Res}_k$$

which is precisely the same form as the equations used for the second-order recursive least-squares filter of the preceding chapter. Therefore, the only possible difference between the second-order recursive least-squares filter and polynomial Kalman filter would be in the gains.

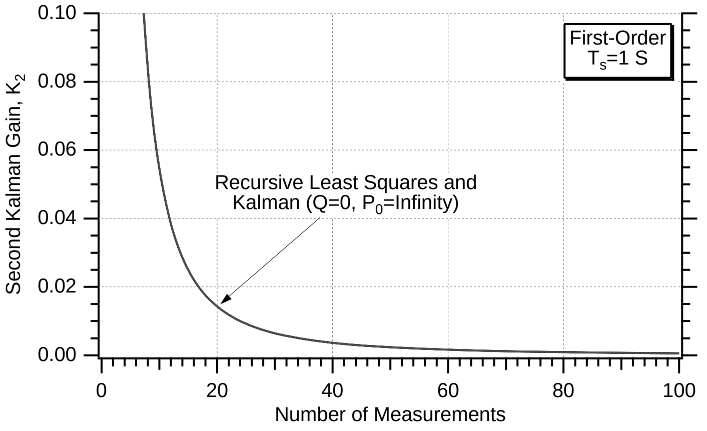

Fig. 4.4 Second gain of first-order polynomial Kalman and recursive least-squares filters are identical.

Next, we would like to compare the gains and variance predictions of both the second-order recursive least-squares filter and the second-order polynomial Kalman filter with zero process-noise and infinite initial covariance matrix. Recall from the preceding chapter that the formulas for the three second-order recursive least-squares filter gains are given by

$$K_{1_k} = \frac{3(3k^2 - 3k + 2)}{k(k+1)(k+2)} \qquad k = 1, 2, \ldots, n$$

$$K_{2k} = \frac{18(2k-1)}{k(k+1)(k+2)T_s}$$

$$K_{3_k} = \frac{60}{k(k+1)(k+2)T_s^2}$$

whereas the formulas for the variance of the errors in the estimates of the first, second, and third states are given by

$$P_{11k} = \frac{3(3k^2 - 3k + 2)\sigma_n^2}{k(k+1)(k+2)}$$

$$P_{22_k} = \frac{12(16K^2 - 30k + 11)\sigma_n^2}{k(k^2-1)(k^2-4)T_s^2}$$

$$P_{33_k} = \frac{720\sigma_n^2}{k(k^2-1)(k^2-4)T_s^4}$$

A program was written and is displayed in Listing 4.2 in which the second-order polynomial Kalman-filter Riccati equations, assuming zero process noise and infinite diagonal elements of the initial covariance matrix, were solved for the gains and covariance matrix. The program included the preceding formulas for the second-order recursive least-squares filter so that both the covariance matrix projections and gain calculations could be compared.

Figure 4.5 compares the square root of the first diagonal element of the second-order polynomial Kalman-filter covariance matrix to the second-order recursive least-squares filter formula for the error in the estimate of the first state. We can see from Fig. 4.5 that the comparison is exact. In addition, this plot also represents how the standard deviation of the error in the estimate of the first state improves as more measurements are taken. In this example the input noise had a standard deviation of unity. After approximately 30 measurements (only 15 measurements for first-order filter) the standard deviation of the error in the estimate is approximately 0.5, meaning that filtering has effectively reduced the effect of noise by a factor of two.

Figure 4.6 compares the square root of the second diagonal element of the second-order polynomial Kalman-filter covariance matrix to the P_{22} formula of the second-order recursive least-squares filter for the error in the estimate of the second state. Again we can see that the comparison is exact. The plot also shows how rapidly the error in the estimate of the second state (i.e., derivative of first state) diminishes after only a few measurements are taken.

Listing 4.2 Comparing second-order Kalman and recursive least-squares filter gains and covariance matrices

```
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 M(3,3),P(3,3),K(3,1),PHI(3,3),H(1,3),R(1,1),PHIT(3,3)
      REAL*8 PHIP(3,3),HT(3,1),KH(3,3),IKH(3,3)
      REAL*8 MHT(3,1),HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),IDN(3,3)
      REAL*8 K1GM,K2GM,K3GM
      INTEGER ORDER
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      ORDER =3
      TS=1.
      SIGNOISE=1.
      DO 1000 I=1,ORDER
      DO 1000 J=1,ORDER
                 PHI(I,J)=0.
                 P(I,J)=0.
                 IDN(I,J)=0.
1000  CONTINUE
      IDN(1,1)=1.
      IDN(2,2)=1.
      IDN(3,3)=1.
      P(1,1)=99999999999999.
      P(2,2)=99999999999999.
      P(3,3)=99999999999999.
      PHI(1,1)=1
      PHI(1,2)=TS
      PHI(1,3)=.5*TS*TS
      PHI(2,2)=1
      PHI(2,3)=TS
      PHI(3,3)=1
      DO 1100 I=1,ORDER
                 H(1,I)=0.
1100  CONTINUE
      H(1,1)=1
      CALL MATTRN(H,1,ORDER,HT)
      R(1,1)=SIGNOISE**2
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      DO 10 XN=1.,100
            CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
            CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,M)
            CALL MATMUL(M,ORDER,ORDER,HT,ORDER,1,MHT)
            CALL MATMUL(H,1,ORDER,MHT,ORDER,1,HMHT)
            HMHTR(1,1)=HMHT(1,1)+R(1,1)
            HMHTRINV(1,1)=1./HMHTR(1,1)
            CALL MATMUL(MHT,ORDER,1,HMHTRINV,1,1,K)
            CALL MATMUL(K,ORDER,1,H,1,ORDER,KH)
            CALL MATSUB(IDN,ORDER,ORDER,KH,IKH)
            CALL MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
            IF(XN<3)THEN
                  P11GM=9999999999.
                  P22GM=9999999999.
```

(*continued*)

Listing 4.2 (*Continued*)

```
                  P33GM=9999999999.
            ELSE
                  P11GM=(3*(3*XN*XN-3*XN+2)/(XN*(XN+1)*(XN+2)))
1                       *SIGNOISE**2
                  P22GM=(12*(16*XN*XN-30*XN+11)/(XN*(XN*XN-1)
1                       *(XN*XN-4)*TS*TS))*SIGNOISE**2
                  P33GM=(720/(XN*(XN*XN-1)*(XN*XN-4)*TS*TS*TS*TS
1                       ))*SIGNOISE**2
            ENDIF
            SP11=SQRT(P(1,1))
            SP22=SQRT(P(2,2))
            SP33=SQRT(P(3,3))
            SP11GM=SQRT(P11GM)
            SP22GM=SQRT(P22GM)
            SP33GM=SQRT(P33GM)
            K1GM=3*(3*XN*XN-3*XN+2)/(XN*(XN+1)*(XN+2))
            K2GM=18*(2*XN-1)/(XN*(XN+1)*(XN+2)*TS)
            K3GM=60/(XN*(XN+1)*(XN+2)*TS*TS)
            IF(XN>=3)THEN
            WRITE(9,*)XN,K(1,1),K1GM,K(2,1),K2GM,K(3,1),K3GM
            WRITE(1,*)XN,K(1,1),K1GM,K(2,1),K2GM,K(3,1),K3GM,
1                 SP11,SP11GM,SP22,SP22GM,SP33,SP33GM

            ENDIF
10    CONTINUE
      PAUSE
      CLOSE(1)
      END

C SUBROUTINE MATTRN IS SHOWN IN LISTING 1.3
C SUBROUTINE MATMUL IS SHOWN IN LISTING 1.4
C SUBROUTINE MATADD IS SHOWN IN LISTING 1.1
C SUBROUTINE MATSUB IS SHOWN IN LISTING 1.2
```

Figure 4.7 compares the square root of the third diagonal element of the second-order polynomial Kalman-filter covariance matrix to the P_{33} formula for the second-order recursive least-squares filter. Again, we can see from Fig. 4.7 that the comparison is exact. The plot also shows how rapidly the error in the estimate of the third state (i.e., second derivative of first state) diminishes after only a few measurements are taken.

Figures 4.8–4.10 compare the gains of both the second-order polynomial Kalman and recursive least-squares filters. We can see that in all cases the comparison is exact. Because the structure and gains of both second-order filters are exactly the same, the filters must be identical. Therefore, we can say once again that simulation experiments have confirmed *that a polynomial Kalman filter with zero process-noise and infinite initial covariance matrix is identical to a recursive least-squares filter of the same order.*

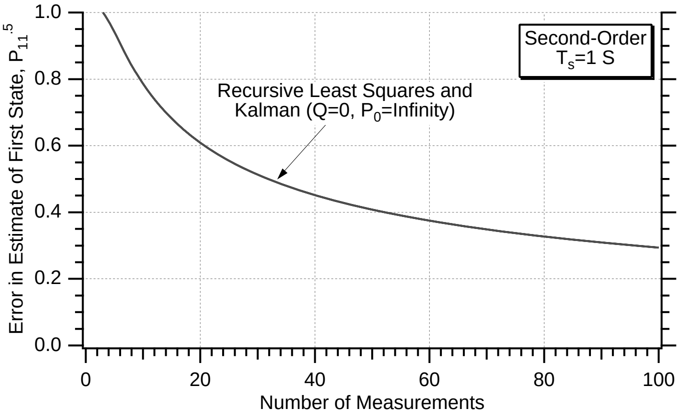

Fig. 4.5 Covariance matrix projections of first state are identical for both second-order polynomial Kalman and recursive least-squares filters.

Comparing Different-Order Filters

We have already demonstrated that the zeroth-order, first-order, and second-order polynomial Kalman and recursive least-squares filters are identical in every way when the Kalman filter has zero process noise and infinite initial covariance matrix. It is now of interest to see how the different-order filters compare in terms of their performance projections. Figure 4.11 compares the standard deviation of the error in the estimate of the first state (i.e., true first state minus estimated first state) of a zeroth, first- and second-order, zero process-noise polynomial Kalman filter as a function of the number of measurements taken. As expected, the errors in the estimates decrease as the number of measurements taken increase. We can

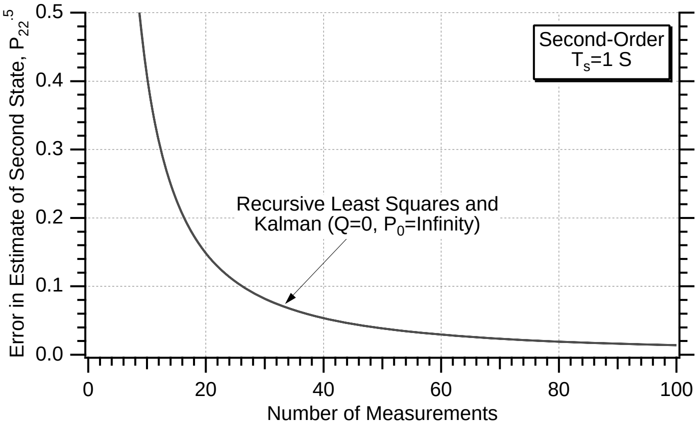

Fig. 4.6 Covariance matrix projections of second state are identical for both second-order polynomial Kalman and recursive least-squares filters.

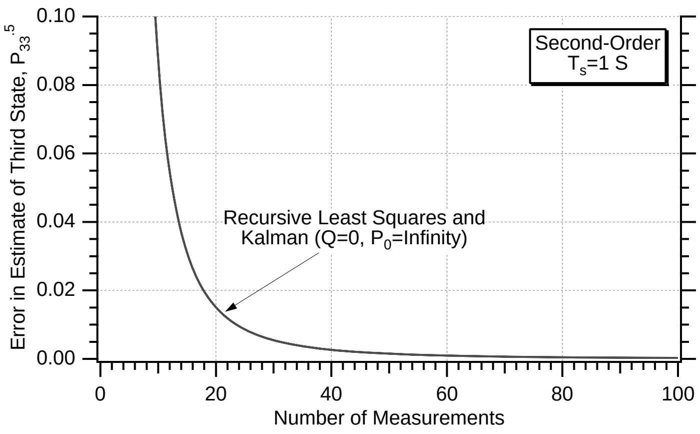

Fig. 4.7 Covariance matrix projections of third state are identical for both second-order polynomial Kalman and recursive least-squares filters.

see that the errors in the estimate are reduced more rapidly for the lower-order filters. This means that maximum noise reduction will be obtained by using the lowest-order filter possible. In practice, truncation error or the ability to track higher-order signals will place the limit on how low an order filter can be used.

Figure 4.12 displays the standard deviation of the error in the estimate of the second state as a function of the number of measurements taken for the first- and second-order, zero process-noise polynomial Kalman filters. The zeroth-order filter was not included because it does not have the ability to estimate the derivative of the first state (i.e., it assumes signal is a constant or of zeroth order and therefore, by definition, has a zero derivative). Again, we can see that the

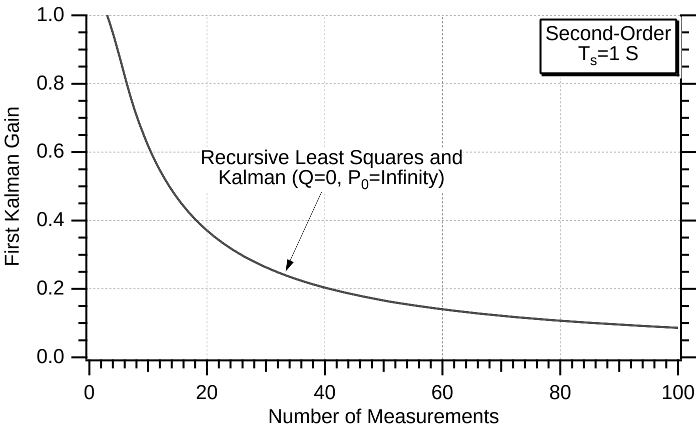

Fig. 4.8 First gain of second-order polynomial Kalman and recursive least-squares filters are identical.

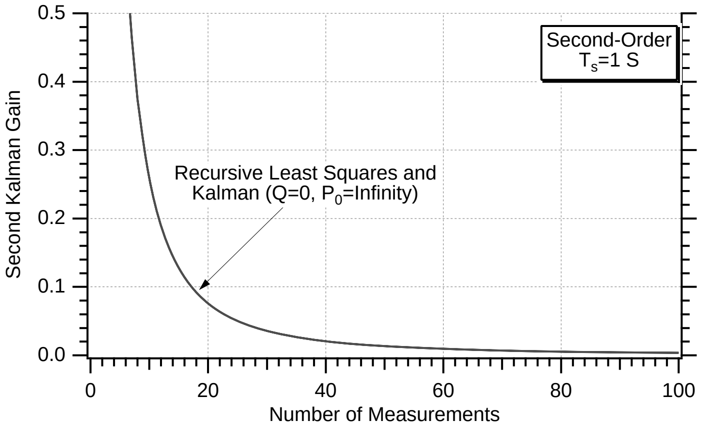

Fig. 4.9 Second gain of second-order polynomial Kalman and recursive least-squares filters are identical.

errors in the estimates decrease as the number of measurements taken increase. As was the case before, the lower-order filter reduces the errors in the estimate of the second state more rapidly.

Finally, Fig. 4.13 displays the standard deviation of the error in the estimate of the third state as a function of the number of measurements taken for the second-order, zero process-noise polynomial Kalman filter. The zeroth-order and first-order filters were not included because they do not have the ability to estimate the derivative of the third state (i.e., second derivative of first state) because they both implicitly assume that the derivative is zero by definition. Again, we can see that the errors in the estimates decrease as the number of measurements taken increase.

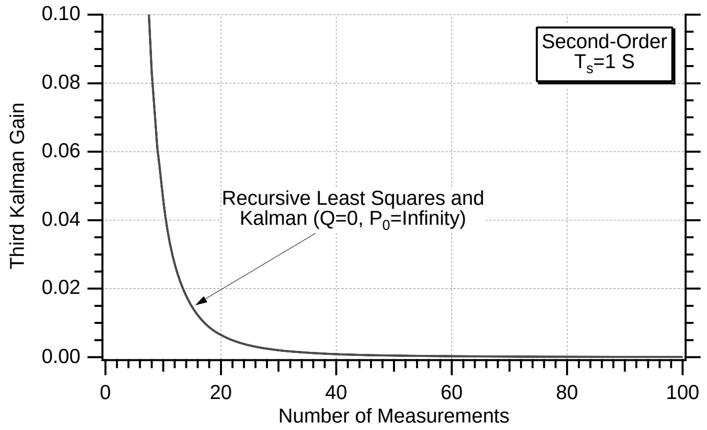

Fig. 4.10 Third gain of second-order polynomial Kalman and recursive least-squares filters are identical.

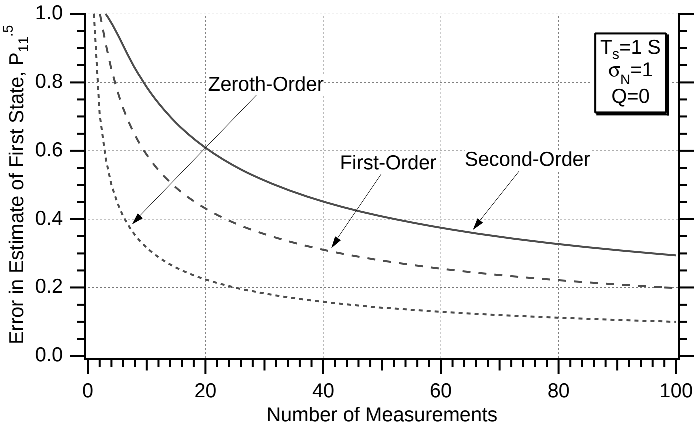

Fig. 4.11 Errors in estimate of first state are smaller with lower-order filters.

Initial Covariance Matrix

So far we have assumed that the polynomial Kalman filter had an initial covariance matrix whose diagonal elements were infinite. This really meant that we had no a priori information on how to initialize the states of the Kalman filter. In practice, information from other sources may be available to help in the initialization process. Under these circumstances the diagonal elements of the initial covariance matrix can be made smaller to reflect more favorable circumstances (i.e., we are not as ignorant).

An experiment was conducted in which the initial covariance matrix (i.e., a number in this example because the matrix is a scalar) for the zeroth-order polynomial Kalman filter was made a parameter. Figure 4.14 shows that the error

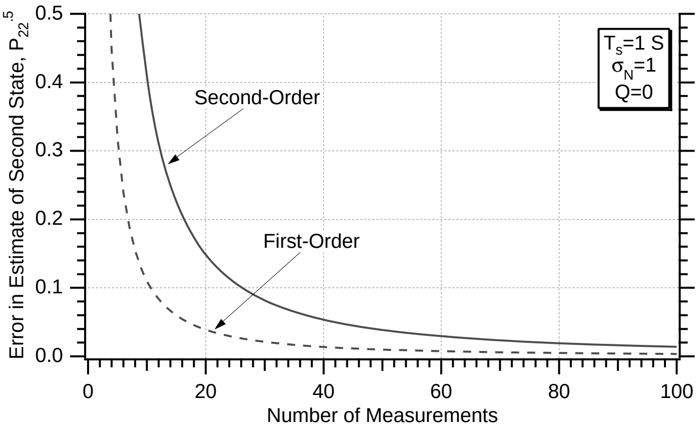

Fig. 4.12 Errors in estimate of second state are smaller with lower-order filter.

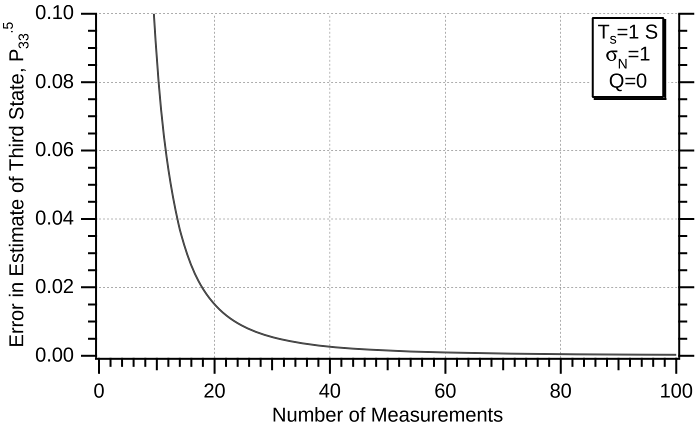

Fig. 4.13 Errors in estimate of third state decrease as number of measurements taken increase.

in the estimate of the first state for the zeroth-order filter does not change as the initial covariance matrix is reduced from infinity to 100. Reducing the initial covariance matrix further by another two orders of magnitude to unity only slightly changes the errors in the estimates when only a few measurements are taken (i.e., less than 20). When many measurements are taken, the answers are the same, regardless of the value for the initial covariance. Reducing the initial covariance matrix by another order of magnitude reduces the errors in the estimates for the first 30 measurements. If the initial covariance matrix is made zero, there will not be any errors in the estimates. Under these circumstances it is assumed that the filter has been perfectly initialized. Because there is no process

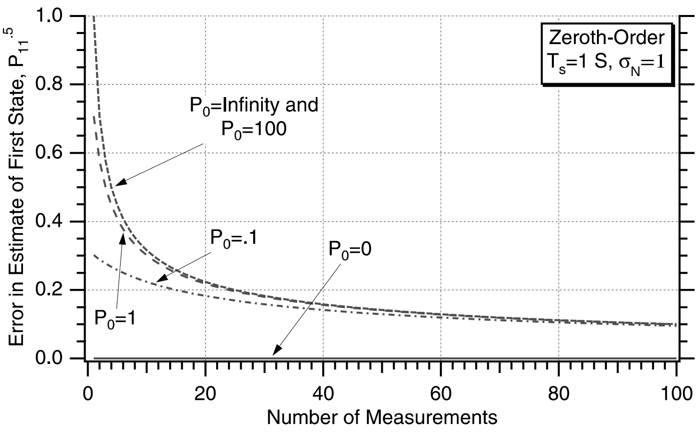

Fig. 4.14 Errors in estimates of first state of a zeroth-order polynomial Kalman filter are fairly insensitive to the initial covariance matrix.

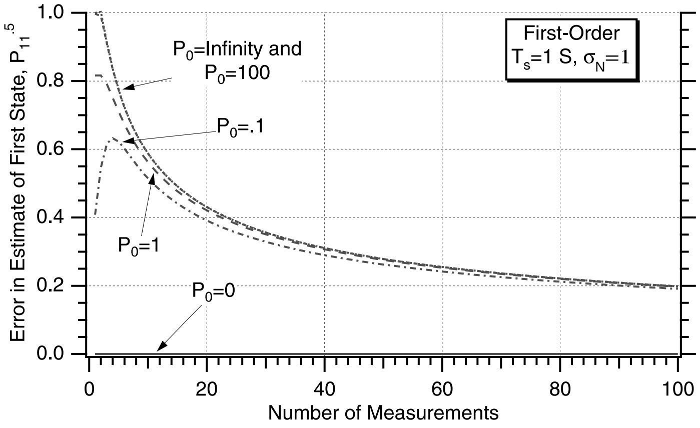

Fig. 4.15 Errors in estimates of first state of a first-order polynomial Kalman filter are fairly insensitive to the initial covariance matrix.

noise, we are assuming that the filter has a perfect model of the real world. Under these circumstances future estimates will be perfect, and there will not be any errors in the estimates.

For the first-order polynomial Kalman filter the initial covariance matrix has two diagonal terms and two off-diagonal terms. The off-diagonal terms were set to zero, whereas the diagonal terms were set equal to each other and reduced from infinity to one tenth. Figures 4.15 and 4.16 show that under these circumstances the error in the estimate of the first and second states of the first-order filter are also fairly insensitive to the initial value of the diagonal terms. Only when the diagonal terms are set to zero do we get significantly different answers. Therefore,

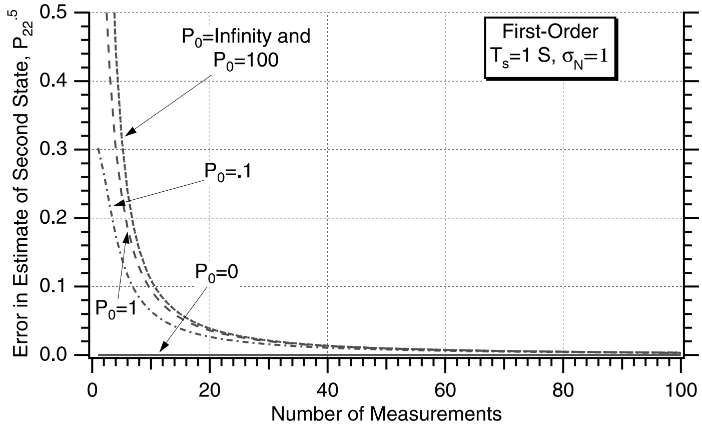

Fig. 4.16 Errors in estimates of second state of a first-order polynomial Kalman filter are fairly insensitive to the initial covariance matrix.

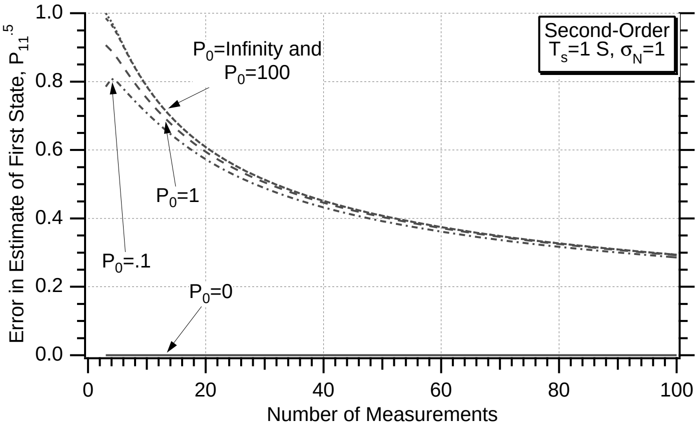

Fig. 4.17 Errors in estimates of first state of a second-order polynomial Kalman filter are fairly insensitive to the initial covariance matrix.

we can conclude that the theoretical performance (i.e., standard deviation of the error in the estimate of the first and second states) of the first-order polynomial Kalman filter is also insensitive to the initial value of the covariance matrix.

For the second-order polynomial Kalman filter the initial covariance matrix has three diagonal terms and six off-diagonal terms (i.e., nine terms in a 3×3 matrix). For this experiment the off-diagonal terms were set to zero while the diagonal terms were set equal to each other and reduced from infinity to one tenth. Figures 4.17–4.19 show that under these circumstances the error in the estimate of the first, second, and third states of the second-order filter is also fairly insensitive to the initial value of the diagonal terms. Again, only when the

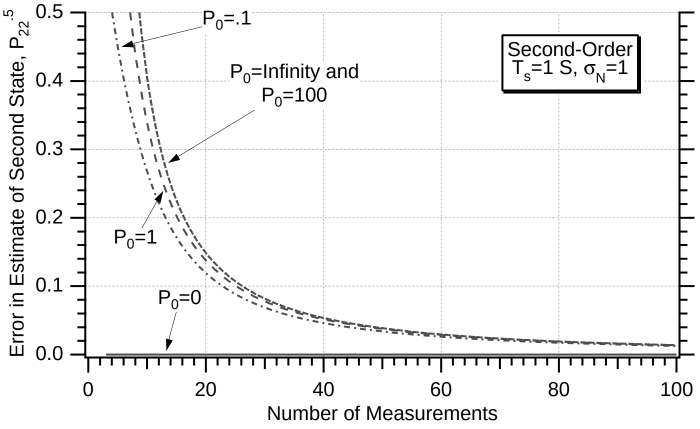

Fig. 4.18 Errors in estimates of second state of a second-order polynomial Kalman filter are fairly insensitive to the initial covariance matrix.

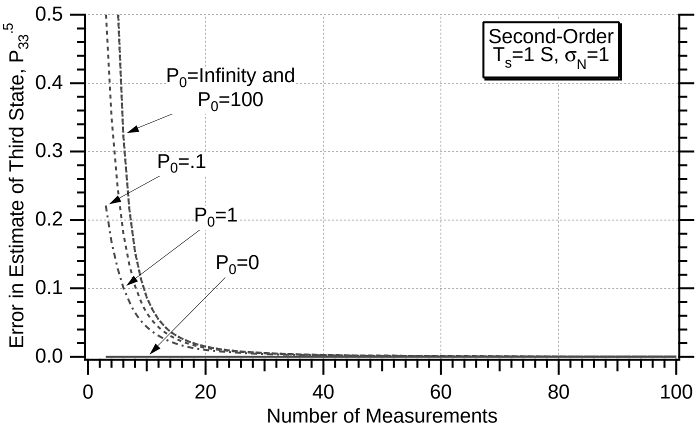

Fig. 4.19 Errors in estimates of third state of a second-order polynomial Kalman filter are fairly insensitive to the initial covariance matrix.

diagonal terms are set to zero do we get significantly different answers. Therefore, we can conclude that the theoretical performance (i.e., standard deviation of the error in the estimate of the first, second, and third states) of the second-order polynomial Kalman filter is also insensitive to the initial value of the covariance matrix.

Riccati Equations with Process Noise

Recall that the Riccati equations are a set of recursive matrix equations given by

$$\boldsymbol{M}_k = \Phi_k \boldsymbol{P}_{k-1} \Phi_k^T + \boldsymbol{Q}_k$$
$$\boldsymbol{K}_k = \boldsymbol{M}_k \boldsymbol{H}^T (\boldsymbol{H} \boldsymbol{M}_k \boldsymbol{H}^T + \boldsymbol{R}_k)^{-1}$$
$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}) \boldsymbol{M}_k$$

where $\boldsymbol{P}_k$ is a covariance matrix representing errors in the state estimates (i.e., diagonal elements represent variance of true state minus estimated state) after an update and $\boldsymbol{M}_k$ is the covariance matrix representing errors in the state estimates before an update. The discrete process-noise matrix $\boldsymbol{Q}_k$, which until now in this chapter was assumed to be zero, can be found from the continuous process-noise matrix $\boldsymbol{Q}$ and the fundamental matrix according to

$$\boldsymbol{Q}_k = \int_0^{T_s} \Phi(\tau) \boldsymbol{Q} \Phi^T(\tau)\, \mathrm{d}t$$

Table 4.2 lists particularly useful $\boldsymbol{Q}$ matrices for different-order polynomial Kalman filters, along with the appropriate fundamental matrices (i.e., derived at the beginning of this chapter) and the derived discrete $\boldsymbol{Q}_k$ matrices. The

Table 4.2 Discrete process-noise matrix varies with system order

Order	Continuous Q	Fundamental	Discrete Q
0	$Q = \Phi_s$	$\Phi_k = 1$	$Q_k = \Phi_s T_s$
1	$Q = \Phi_s \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$	$\Phi_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$	$Q_k = \Phi_s \begin{bmatrix} \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^2}{2} & T_s \end{bmatrix}$
2	$Q = \Phi_s \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$\Phi_k = \begin{bmatrix} 1 & T_s & 0.5T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}$	$Q_k = \Phi_s \begin{bmatrix} \frac{T_s^5}{20} & \frac{T_s^4}{8} & \frac{T_s^3}{6} \\ \frac{T_s^4}{8} & \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^3}{6} & \frac{T_s^2}{2} & T_s \end{bmatrix}$

continuous process-noise matrix assumes that the process noise always enters the system model on the derivative of the last state (i.e., highest derivative). We can see that the discrete process-noise matrix depends on the continuous process-noise spectral density Φ_s and the sampling time T_s. When the continuous process-noise spectral density Φ_s does not represent actual model noise uncertainty, it simply becomes a fudge factor that accounts for our lack of knowledge of the real world. If Φ_s is zero or small, it means that we believe our Kalman filter model of the real world is excellent. Large values of Φ_s mean that we really do not believe that the Kalman filter has a good model of the real world and the filter must be prepared for a great deal of uncertainty. Usually Φ_s is determined experimentally by performing many Monte Carlo experiments under a variety of scenarios.

Let us first examine how the zeroth-order Kalman filter's ability to estimate is influenced by process noise. Figure 4.20 shows how the error in the estimate of the state (i.e., true state minus estimated state) worsens as more process noise is added to the zeroth-order filter for the case in which the standard deviation of the measurement noise is unity and the sampling time is 1 s. We can see from Fig. 4.20 that when the process noise is zero the error in the estimate will eventually approach zero as more measurements are taken. However, when process noise is present, the error in the estimate will approach a steady-state value after only a few measurements, and the quality of the estimates will not improve beyond a certain point no matter how many more measurements are taken. Increasing the value of the process noise increases the error in the estimate because the filter is assuming that process noise is continually contributing uncertainty to the states even when the actual process noise does not exist.

Figures 4.21 and 4.22 show how the error in the estimate of the two states (i.e., true state minus estimated state) of the first-order polynomial Kalman filter increases as more process noise is added for the case in which the standard

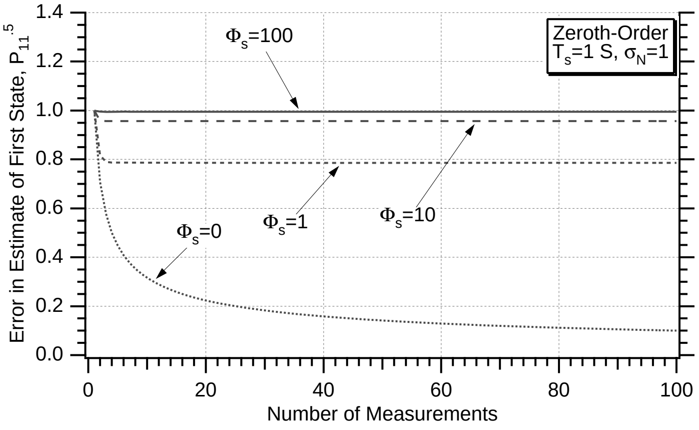

Fig. 4.20 Error in estimate of the state degrades as process noise increases for zeroth-order filter.

deviation of the measurement noise is unity and the sampling time is 1 s. As was the case earlier, we can see from Figs. 4.21 and 4.22 that when the process noise is zero the error in the state estimate will eventually approach zero as more measurements are taken. However, when process noise is present, the error in the state estimates will approach a steady-state value after only a few measurements are taken. As was the case earlier, increasing the value of the process noise increases the error in the estimate.

Figures 4.23–4.25 show how the error in the estimate of the three states (i.e., true state minus estimated state) of the second-order polynomial Kalman filter increase as more process noise is added for the case in which the standard

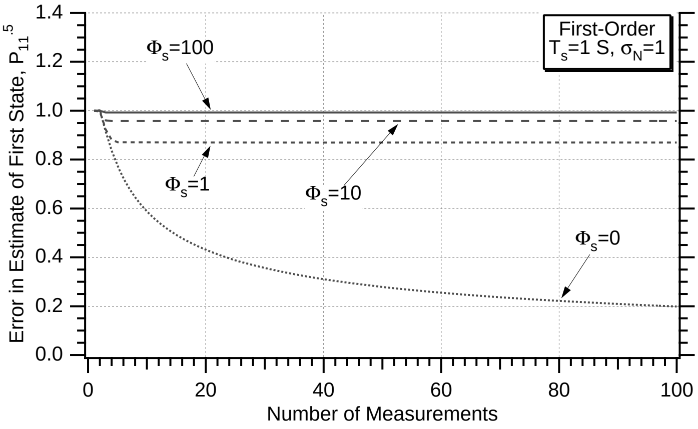

Fig. 4.21 Increasing the process noise increases the error in the estimate of the first state of a first-order polynomial Kalman filter.

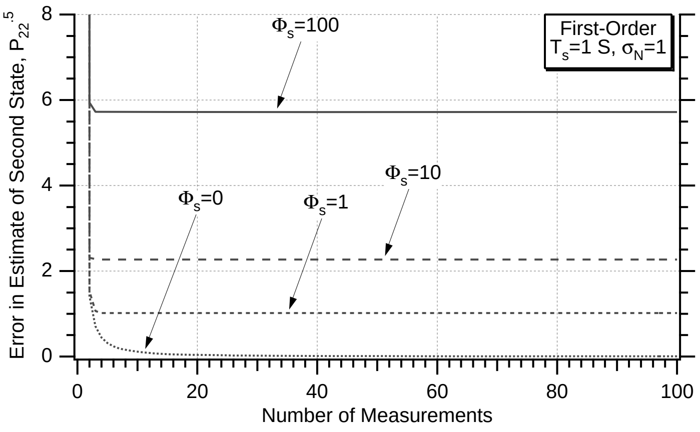

Fig. 4.22 **Increasing the process noise increases the error in the estimate of the second state of a first-order polynomial Kalman filter.**

deviation of the measurement noise is unity and the sampling time is 1 s. As was the case for both the zeroth- and first-order polynomial Kalman filters, we can see from Figs. 4.23–4.25 that when the process noise is zero the error in the state estimate will eventually approach zero as more measurements are taken. Again, when process noise is present, the error in the state estimates will approach a steady-state value after only a few measurements are taken. As was the case for the zeroth- and first-order polynomial Kalman filters, increasing the value of the process noise also increases the error in the estimate for the second-order polynomial Kalman filter.

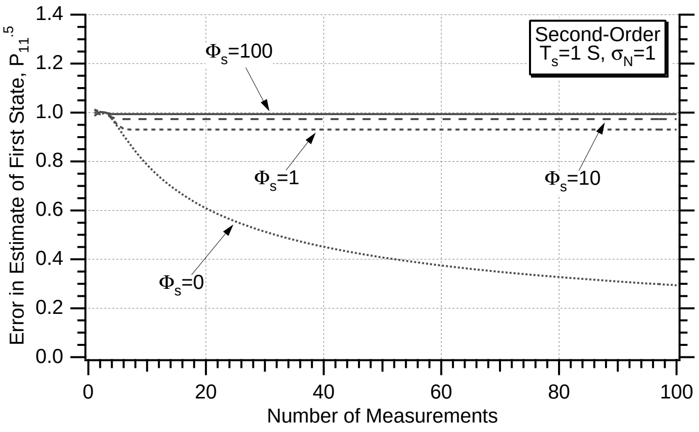

Fig. 4.23 **Increasing the process noise increases the error in the estimate of the first state of a second-order polynomial Kalman filter.**

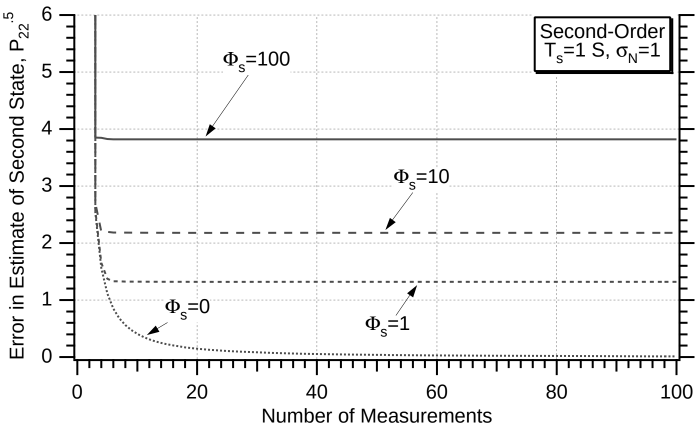

Fig. 4.24 Increasing the process noise increases the error in the estimate of the second state of a second-order polynomial Kalman filter.

Example of Kalman Filter Tracking a Falling Object

To illustrate the utility of a polynomial Kalman filter with process noise, let us consider the one-dimensional example of an object falling rather quickly on a tracking radar, as shown in Fig. 4.26. The object is initially 400,000 ft above the radar and has a velocity of 6000 ft/s toward the radar, which is located on the surface of a flat Earth. In this example we are neglecting drag or air resistance so that only gravity g (i.e., $g = 32.2$ ft/s^2) acts on the object. Let us pretend that the radar measures the range from the radar to the target (i.e., altitude of the target) with a 1000-ft standard deviation measurement accuracy. The radar takes measurement 10 times a second for 30 s. We would like to build a filter to

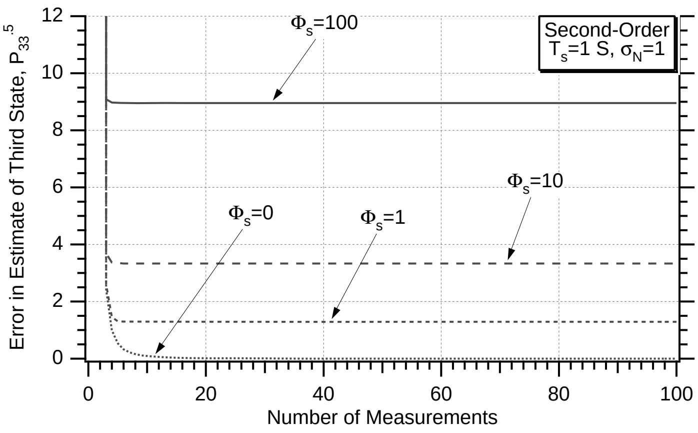

Fig. 4.25 Increasing the process noise increases the error in the estimate of the third state of a second-order polynomial Kalman filter.

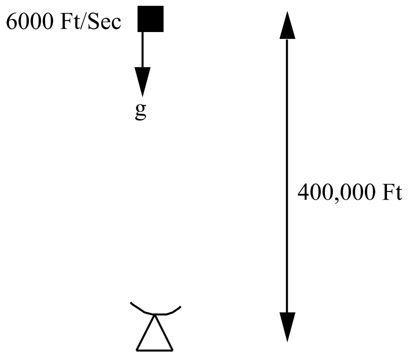

Fig. 4.26 Radar tracking falling object.

estimate the altitude and velocity of the object without any a priori information (i.e., knowing initial altitude and velocity of the object).

We know from basic physics that if x is the distance from the radar to the object, then the value of x at any time t is given by

$$x = 400{,}000 - 6000t - \frac{gt^2}{2}$$

As a check, we set time to zero in the preceding expression and get the correct initial value for x of 400,000 ft. The velocity of the object at any time can be found by taking the derivative of the preceding expression, yielding

$$\dot{x} = -6000 - gt$$

Again, as a check we set time to zero in the preceding expression and get the correct initial velocity of −6000 ft/s. We can see that the expression for altitude x is a second-order polynomial in time. Because, in this example, the real world is actually a second-order polynomial, a second-order polynomial Kalman filter with zero process noise can be used to track the object.

Listing 4.3 presents the simulation that will be used to illustrate how the second-order polynomial Kalman filter's processing of the radar measurements can be used to estimate the altitude and velocity of the target. We can see from Listing 4.3 that the filter states are initialized to zero (i.e., XH = XDH = XDDH = 0) and that the diagonal elements of the initial covariance matrix are infinite [i.e., P(1,1) = P(2,2) = P(3,3) = 9999999999999] because we do not have any a priori information concerning the initial target altitude or speed. The initial state estimates are in considerable error because the initial altitude is 400,000 ft. The initial velocity is −6000 ft/s, and the acceleration is always -32.2 ft/s^2. Other parameters have been set in accordance with the problem statement. We print out the filter state estimates along with the true states. In addition, we compute the actual error in the state estimates along with the theoretical predictions from the diagonal elements of the covariance matrix.

Figure 4.27 displays the actual altitude of the object along with the filter's estimate of altitude as a function of time. We can see that the filter is doing an

Listing 4.3 Simulation of radar tracking falling object

```
C THE FIRST THREE STATEMENTS INVOKE THE ABSOFT RANDOM
  NUMBER GENERATOR ON THE MACINTOSH
      GLOBAL DEFINE
            INCLUDE 'quickdraw.inc'
      END
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 M(3,3),P(3,3),K(3,1),PHI(3,3),H(1,3),R(1,1),PHIT(3,3)
      REAL*8 PHIP(3,3),HT(3,1),KH(3,3),IKH(3,3)
      REAL*8 MHT(3,1),HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),IDN(3,3)
      REAL*8 Q(3,3),PHIPPHIT(3,3)
      INTEGER ORDER
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      ORDER =3
      PHIS=0.
      TS=.1
      A0=400000
      A1=-6000.
      A2=-16.1
      XH=0
      XDH=0
      XDDH=0
      SIGNOISE=1000.
      DO 1000 I=1,ORDER
      DO 1000 J=1,ORDER
            PHI(I,J)=0.
            P(I,J)=0.
            IDN(I,J)=0.
            Q(I,J)=0.
1000  CONTINUE
      IDN(1,1)=1.
      IDN(2,2)=1.
      IDN(3,3)=1.
      P(1,1)=99999999999999.
      P(2,2)=99999999999999.
      P(3,3)=99999999999999.
      PHI(1,1)=1
      PHI(1,2)=TS
      PHI(1,3)=.5*TS*TS
      PHI(2,2)=1
      PHI(2,3)=TS
      PHI(3,3)=1
      DO 1100 I=1,ORDER
            H(1,I)=0.
1100  CONTINUE
      H(1,1)=1
      CALL MATTRN(H,1,ORDER,HT)
      R(1,1)=SIGNOISE**2
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      Q(1,1)=PHIS*TS**5/20
```

(*continued*)

Listing 4.3 (*Continued*)

```
      Q(1,2)=PHIS*TS**4/8
      Q(1,3)=PHIS*TS**3/6
      Q(2,1)=Q(1,2)
      Q(2,2)=PHIS*TS**3/3
      Q(2,3)=PHIS*TS*TS/2
      Q(3,1)=Q(1,3)
      Q(3,2)=Q(2,3)
      Q(3,3)=PHIS*TS
      DO 10 T=0.,30.,TS
         CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
         CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,
              PHIPPHIT)
         CALL MATADD(PHIPPHIT,ORDER,ORDER,Q,M)
         CALL MATMUL(M,ORDER,ORDER,HT,ORDER,1,MHT)
         CALL MATMUL(H,1,ORDER,MHT,ORDER,1,HMHT)
         HMHTR(1,1)=HMHT(1,1)+R(1,1)
         HMHTRINV(1,1)=1./HMHTR(1,1)
         CALL MATMUL(MHT,ORDER,1,HMHTRINV,1,1,K)
         CALL MATMUL(K,ORDER,1,H,1,ORDER,KH)
         CALL MATSUB(IDN,ORDER,ORDER,KH,IKH)
         CALL MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
         CALL GAUSS(XNOISE,SIGNOISE)
         X=A0+A1*T+A2*T*T
         XD=A1+2*A2*T
         XDD=2*A2
         XS=X+XNOISE
         RES=XS-XH-TS*XDH-.5*TS*TS*XDDH
         XH=XH+XDH*TS+.5*TS*TS*XDDH+K(1,1)*RES
         XDH=XDH+XDDH*TS+K(2,1)*RES
         XDDH=XDDH+K(3,1)*RES
         SP11=SQRT(P(1,1))
         SP22=SQRT(P(2,2))
         SP33=SQRT(P(3,3))
         XHERR=X-XH
         XDHERR=XD-XDH
         XDDHERR=XDD-XDDH
         WRITE(9,*)T,X,XH,XD,XDH,XDD,XDDH
         WRITE(1,*)T,X,XH,XD,XDH,XDD,XDDH
         WRITE(2,*)T,XHERR,SP11,-SP11,XDHERR,SP22,-SP22,XDDHERR,
1        SP33,-SP33
10    CONTINUE
      CLOSE(1)
      CLOSE(2)
      PAUSE
      END

C SUBROUTINE GAUSS IS SHOWN IN LISTING 1.8
C SUBROUTINE MATTRN IS SHOWN IN LISTING 1.3
C SUBROUTINE MATMUL IS SHOWN IN LISTING 1.4
C SUBROUTINE MATADD IS SHOWN IN LISTING 1.1
C SUBROUTINE MATSUB IS SHOWN IN LISTING 1.2
```

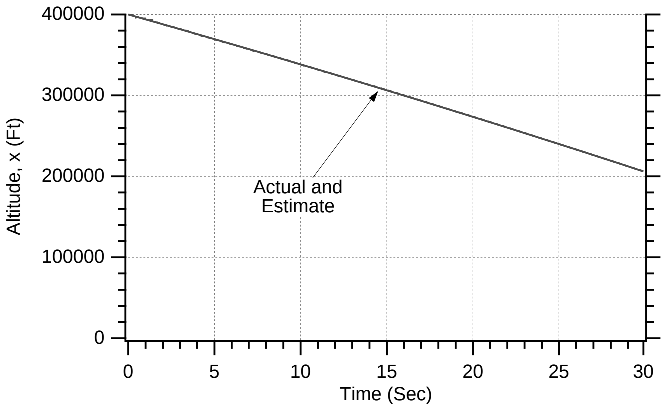

Fig. 4.27 Altitude estimate of falling object is near perfect.

excellent job because the estimate of the object's altitude appears to be virtually exact.

Figure 4.28 displays the actual velocity of the object along with the filter's estimate of velocity. We can see that it takes approximately 10 s for the filter to obtain a highly accurate estimate of the target velocity. The large excursions in the velocity estimate at the beginning of the measurement and estimation process is because the initial estimate of target velocity was 0 ft/s, whereas the actual target velocity was −6000 ft/s. This large initialization error was the main reason it took the filter 10 s to establish an accurate velocity track.

Although we really do not have to estimate the object's acceleration because we know it must be 32.2 ft/s^2 (i.e., only gravity acts on the object), the acceleration estimate is free with a second-order polynomial Kalman filter. Figure 4.29 shows that for nearly 10 s the filter's acceleration estimate is terrible

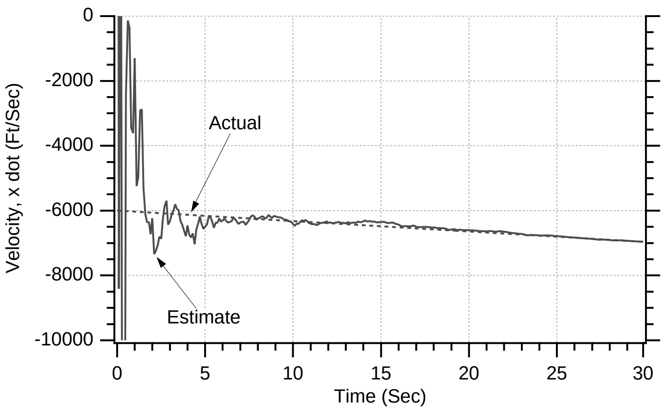

Fig. 4.28 It takes approximately 10 s to accurately estimate velocity of falling object with second-order filter.

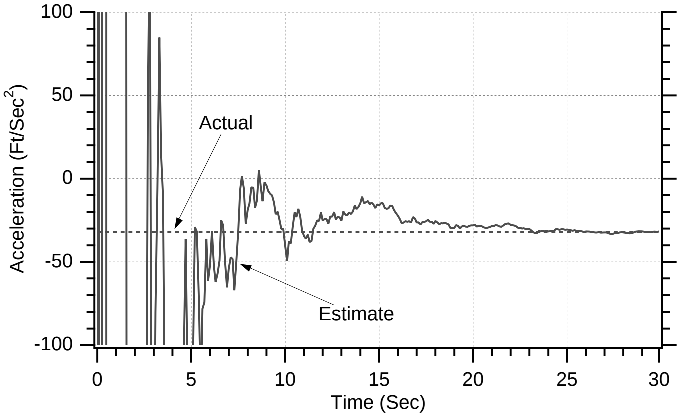

Fig. 4.29 It takes nearly 20 s to accurately estimate acceleration of falling object with second-order filter.

(i.e., because the filter had terrible initial conditions). Only after 20 s do we receive accurate acceleration estimates. Again, the filter had a bad initial estimate of acceleration because it assumed zero, whereas the actual acceleration of the object was -32.2 ft/s^2. Later in this chapter we will see if total filter performance could be improved if we made use of the fact that we know in advance the exact acceleration acting on the object.

As we saw in Chapter 3, just comparing the state estimates to the actual states is not sufficient for establishing that the filter is working according to theory. To be sure that the filter is working properly, we must look at the actual errors in the estimates and compare them to the theoretical answers obtained from the covariance matrix (i.e., square root of first diagonal element for first state, square root of second diagonal element for second state, etc.). Figures 4.30

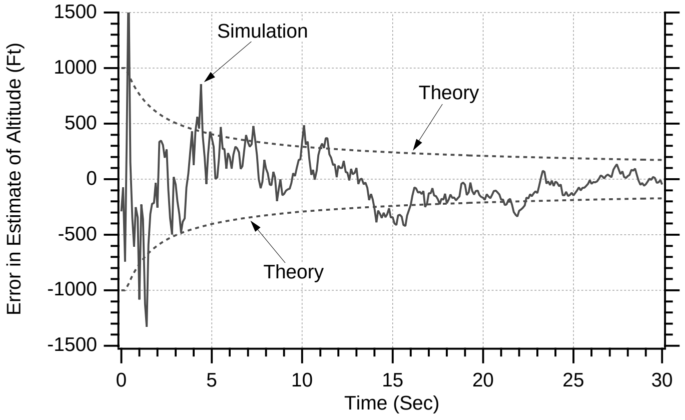

Fig. 4.30 Second-order polynomial Kalman filter single flight results appear to match theory for errors in estimate of altitude.

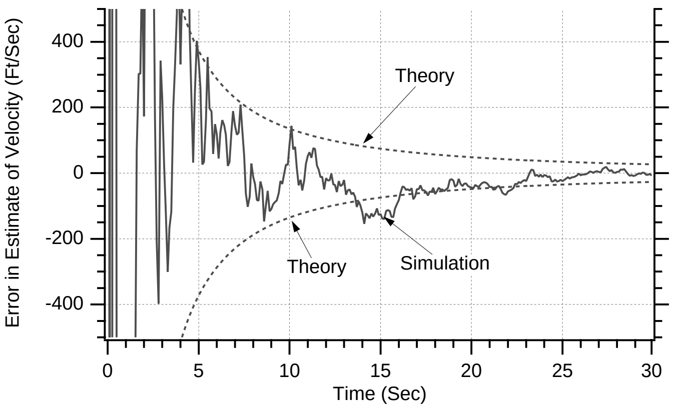

Fig. 4.31 Second-order polynomial Kalman filter single flight results appear to match theory for errors in estimate of velocity.

and 4.31 show that the simulated errors in the estimate of the first and second states lie within the theoretical bounds approximately 68% of the time. In other words, the second-order polynomial Kalman filter appears to be working correctly.

If we were not satisfied with the estimates from the second-order polynomial Kalman filter, we could try a first-order filter. Based on work in this chapter and the preceding chapter, we know that a lower-order filter will have superior noise-reduction properties. However, if we use a first-order filter with zero process noise, Fig. 4.32 shows that our position estimate diverges from the truth because of the gravity term. Filter divergence is expected because we have a first-order filter operating in a second-order world due to gravity.

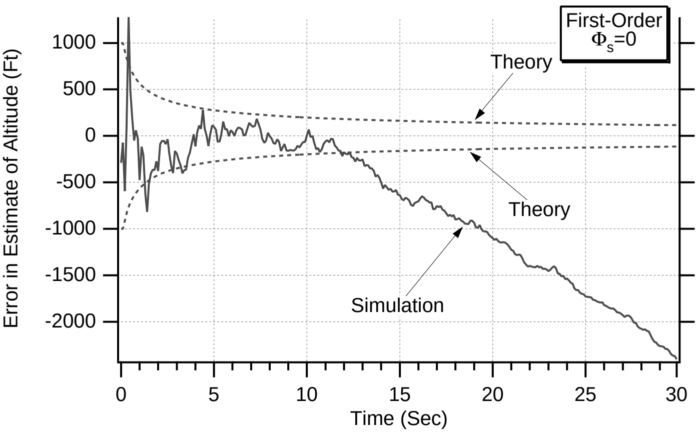

Fig. 4.32 First-order polynomial Kalman filter without process noise can not track second-order signal.

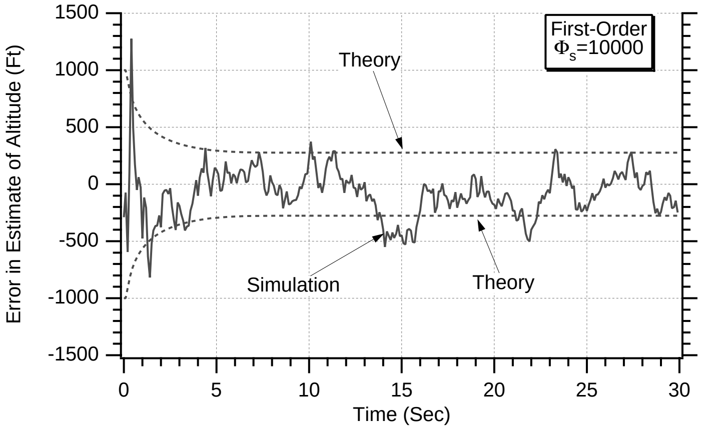

Fig. 4.33 Adding process noise prevents altitude errors of first-order filter from diverging.

One engineering fix to the divergence problem is the use of process noise. Figures 4.33 and 4.34 show that the errors in the estimates of the first-order polynomial Kalman filter can be kept from diverging if we add process noise (i.e., $\Phi_s = 10{,}000$). However, although the addition of process noise to the first-order filter prevents divergence, it also increases the errors in the estimate to the point where we would have been better off using the second-order polynomial Kalman filter without process noise. For example, Fig. 4.33 shows the error in the estimate of altitude for the first-order polynomial Kalman filter with process noise approaches 300 ft in the steady state, whereas Fig. 4.30 shows that the error in the estimate of altitude for the second-order polynomial Kalman filter without process noise approaches 200 ft in the steady state. Figure 4.34 shows that the

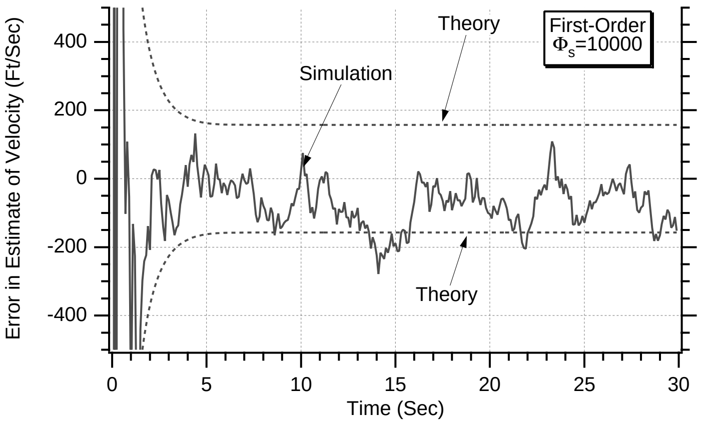

Fig. 4.34 Adding process noise prevents velocity errors of first-order filter from diverging.

error in the estimate of velocity for the first-order polynomial Kalman filter with process noise approaches 150 ft/s in the steady state, whereas Fig. 4.31 shows that the error in the estimate of velocity for the second-order polynomial Kalman filter without process noise approaches only 25 ft/s in the steady state.

We can reduce the estimation errors with a first-order filter even further and without the use of process noise by making use of a priori information. We know that only gravity (i.e., $g = 32.2$ ft/s^2) acts on the falling body. Gravitational information can be incorporated into the Kalman filter by first recalling that if we have a priori deterministic information the real world can be described in state-space form by

$$\dot{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{x} + \boldsymbol{G}\boldsymbol{u} + \boldsymbol{w}$$

With the preceding equation representing the real world the resultant Kalman-filter equation will be

$$\hat{\boldsymbol{x}}_k = \Phi_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{G}_k \boldsymbol{u}_{k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\Phi_k \hat{\boldsymbol{x}}_{k-1} - \boldsymbol{H}\boldsymbol{G}_k \boldsymbol{u}_{k-1})$$

If $\boldsymbol{u}_{k-1}$ is assumed to be constant between sampling instants, $\boldsymbol{G}_k$ is obtained from

$$\boldsymbol{G}_k = \int_0^{T_s} \Phi(\tau)\boldsymbol{G}\,\mathrm{d}\tau$$

For our particular problem we have already pointed out that gravity is known and does not have to be estimated. Therefore, for our particular problem the only differential equation describing the real world is

$$\ddot{x} = -g$$

The preceding second-order differential equation can be recast in state-space form as

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} g$$

From the preceding matrix differential equation we recognize that the systems dynamics matrix is given by

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

For the preceding system dynamics matrix we have already shown that the fundamental matrix turns out to be

$$\Phi_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$$

From the state-space equation we can also see that

$$\boldsymbol{G} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

and that

$$\boldsymbol{u}_{k-1} = g$$

Therefore, the discrete matrix $\boldsymbol{G}_k$ can be found to be

$$\boldsymbol{G}_k = \int_0^{T_s} \Phi(\tau)\boldsymbol{G}\, \mathrm{d}\tau = \int_0^{T_s} \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} \mathrm{d}\tau = \begin{bmatrix} \dfrac{-T_s^2}{2} \\ -T_s \end{bmatrix}$$

Because the formula for the Kalman filter is

$$\hat{\boldsymbol{x}}_k = \Phi_k \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{G}_k \boldsymbol{u}_{k-1} + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}\Phi_k \hat{\boldsymbol{x}}_{k-1} - \boldsymbol{H}\boldsymbol{G}_k \boldsymbol{u}_{k-1})$$

substitution yields

$$\begin{bmatrix} \hat{x}_k \\ \hat{\dot{x}}_k \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \end{bmatrix} + \begin{bmatrix} -0.5T_s^2 \\ -T_s \end{bmatrix} g$$
$$+ \begin{bmatrix} K_{1_k} \\ K_{2_k} \end{bmatrix} \left[x_k^* - [1 \quad 0] \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \end{bmatrix} - [1 \quad 0] \begin{bmatrix} -0.5T_s^2 \\ -T_s \end{bmatrix} g \right]$$

Multiplying out the terms of the preceding equation yields two scalar equations:

$$\hat{x}_k = \hat{x}_{k-1} + \hat{\dot{x}}_{k-1}T_s - 0.5gT_s^2 + K_{1_k}(x_k^* - \hat{x}_{k-1} - \hat{\dot{x}}_{k-1}T_s + 0.5gT_s^2)$$
$$\hat{\dot{x}}_k = \hat{\dot{x}}_{k-1} - gT_s + K_{2_k}(x_k^* - \hat{x}_{k-1} - \hat{\dot{x}}_{k-1}T_s + 0.5gT_s^2)$$

If we define the residual as

$$\mathrm{Res}_k = x_k^* - \hat{x}_{k-1} - \hat{\dot{x}}_{k-1}T_s + 0.5gT_s^2$$

then the two equations for the Kalman filter simplify to

$$\hat{x}_k = \hat{x}_{k-1} + \hat{\dot{x}}_{k-1}T_s - 0.5gT_s^2 + K_{1_k}\mathrm{Res}_k$$
$$\hat{\dot{x}}_k = \hat{\dot{x}}_{k-1} - gT_s + K_{2_k}\mathrm{Res}_k$$

The a priori information on gravity only affects the structure of the Kalman filter. Gravity does not influence the Riccati equations.

Listing 4.4 is a simulation of the falling object being tracked and estimated with the preceding first-order polynomial Kalman filter that compensates for gravity without using process noise. The errors in the estimate of altitude and

Listing 4.4 First-order polynomial Kalman filter with gravity compensation

```
C THE FIRST THREE STATEMENTS INVOKE THE ABSOFT RANDOM
  NUMBER GENERATOR ON THE MACINTOSH
      GLOBAL DEFINE
              INCLUDE 'quickdraw.inc'
      END
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 P(2,2),Q(2,2),M(2,2),PHI(2,2),HMAT(1,2),HT(2,1),PHIT(2,2)
      REAL*8 RMAT(1,1),IDN(2,2),PHIP(2,2),PHIPPHIT(2,2),HM(1,2)
      REAL*8 HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),MHT(2,1),K(2,1)
      REAL*8 KH(2,2),IKH(2,2)
      INTEGER STEP,ORDER
      TS=.1
      PHIS=0.
      A0=400000.
      A1=-6000.
      A2=-16.1
      XH=0.
      XDH=0.
      SIGNOISE=1000.
      ORDER=2
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      T=0.
      S=0.
      H=.001
      DO 14 I=1,ORDER
      DO 14 J=1,ORDER
      PHI(I,J)=0.
      P(I,J)=0.
      Q(I,J)=0.
      IDN(I,J)=0.
14    CONTINUE
      RMAT(1,1)=SIGNOISE**2
      IDN(1,1)=1.
      IDN(2,2)=1.
      P(1,1)=99999999999.
      P(22)=99999999999.
      PHI(1,1)=1.
      PHI(1,2)=TS
      PHI(2,2)=1.
      Q(1,1)=TS*TS*TS*PHIS/3.
      Q(1,2)=.5*TS*TS*PHIS
      Q(2,1)=Q(1,2)
      Q(2,2)=PHIS*TS
      HMAT(1,1)=1.
      HMAT(1,2)=0.
      DO 10 T=0.,30.,TS
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      CALL MATTRN(HMAT,1,ORDER,HT)
```

(*continued*)

Listing 4.4 (*Continued*)

```
      CALL  MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
      CALL  MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,PHIPPHIT)
      CALL  MATADD(PHIPPHIT,ORDER,ORDER,Q,M)
      CALL  MATMUL(HMAT,1,ORDER,M,ORDER,ORDER,HM)
      CALL  MATMUL(HM,1,ORDER,HT,ORDER,1,HMHT)
      CALL  MATADD(HMHT,ORDER,ORDER,RMAT,HMHTR)
            HMHTRINV(1,1)=1./HMHTR(1,1)
      CALL  MATMUL(M,ORDER,ORDER,HT,ORDER,1,MHT)
      CALL  MATMUL(MHT,ORDER,1,HMHTRINV,1,1,K)
      CALL  MATMUL(K,ORDER,1,HMAT,1,ORDER,KH)
      CALL  MATSUB(IDN,ORDER,ORDER,KH,IKH)
      CALL  MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
      CALL  GAUSS(XNOISE,SIGNOISE)
      X=A0+A1*T+A2*T*T
      XD=A1+2*A2*T
      XS=X+XNOISE
      RES=XS-XH-TS*XDH+16.1*TS*TS
      XH=XH+XDH*TS-16.1*TS*TS+K(1,1)*RES
      XDH=XDH-32.2*TS+K(2,1)*RES
      SP11=SQRT(P(1,1))
      SP22=SQRT(P(2,2))
      XHERR=X-XH
      XDHERR=XD-XDH
      WRITE(9,*)T,XD,XDH,K(1,1),K(2,1)
      WRITE(1,*)T,X,XH,XD,XDH
      WRITE(2,*)T,XHERR,SP11,-SP11,XDHERR,SP22,-SP22
10    CONTINUE
      PAUSE
      CLOSE(1)
      END

C SBROUTINE GAUSS IS SHOWN IN LISTING 1.8
C SUBROUTINE MATTRN IS SHOWN IN LISTING 1.3
C SUBROUTINE MATMUL IS SHOWN IN LISTING 1.4
C SUBROUTINE MATADD IS SHOWN IN LISTING 1.1
C SUBROUTINE MATSUB IS SHOWN IN LISTING 1.2
```

velocity, along with the theoretical covariance matrix projections, are printed out every tenth of a second.

Figures 4.35 and 4.36 show that the first-order polynomial Kalman filter with gravity compensation and without process noise is superior to the second-order polynomial Kalman filter without process noise. Figure 4.35 shows that the error in the estimate of altitude has been reduced from 200 ft (i.e., see Fig. 4.30) to 100 ft. Figure 4.36 shows that the error in the estimate of velocity has been reduced from 25 ft/s (i.e., see Fig. 4.31) to less than 5 ft/s.

Thus, we can see that if a priori information is available it pays to use it. A priori information may allow us to use lower-order filters that will in turn reduce the errors in all state estimates.

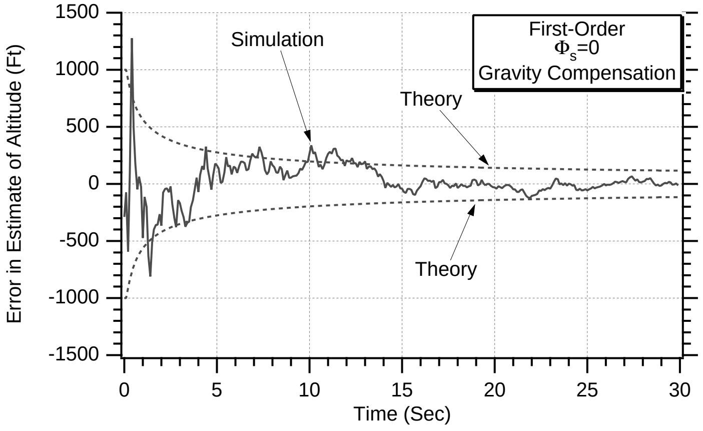

Fig. 4.35 Adding gravity compensation to first-order filter without process noise reduces altitude errors from that of a second-order filter without process noise.

Revisiting Accelerometer Testing Example

To demonstrate another example of the application of polynomial Kalman filters, let us revisit the accelerometer testing example of Chapter 2. Recall that in this example accelerometer measurements are taken for different accelerometer orientation angles, as shown in Fig. 4.37. When the accelerometer input axis is vertical, the accelerometer reading will be g. As the accelerometer input axis rotates through different angles θ_k the accelerometer reading will be $g\cos\theta_k$.

We showed in Chapter 2 that the accelerometer output will not only consist of the gravity term but also the accelerometer bias, scale-factor, and g-sensitive

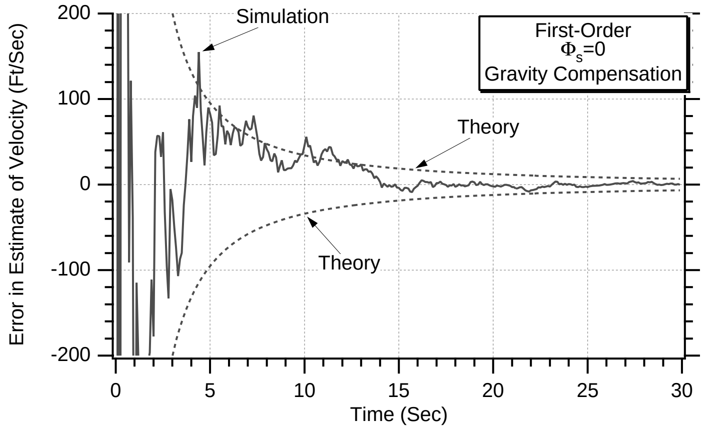

Fig. 4.36 Adding gravity compensation to first-order filter without process noise reduces velocity errors from that of a second-order filter without process noise.

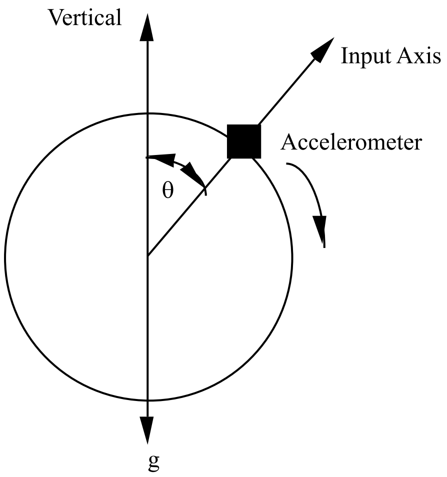

Fig. 4.37 Accelerometer experiment test setup.

drift errors. Therefore, the total accelerometer output is given by

$$\text{Accelerometer Output} = g\cos\theta_k + B + SFg\cos\theta_k + K(g\cos\theta_k)^2$$

where θ_k is the angle the accelerometer makes with the vertical for the kth measurement, g is gravity, B is an accelerometer bias, SF is the accelerometer scale-factor error, and K is a gravity squared or g-squared sensitive drift. Because the theoretical accelerometer output does not have bias, scale-factor, or g-sensitive errors, we expect the output to only contain the vertical angle dependent gravity term or

$$\text{Theory} = g\cos\theta_k$$

Therefore, the accelerometer error is simply the difference between the actual output and theory or

$$\text{Error} = \text{Accelerometer Output} - \text{Theory} = B + SFg\cos\theta_k + K(g\cos\theta_k)^2$$

From a Kalman-filtering point of view, we would like to treat the preceding error equation as measurements at different angles. In addition, we would like to be able to estimate the bias, scale-factor, and g-sensitive drift errors. We also assumed that our measurement of the angle θ_k may not be precise but is corrupted by zero mean Gaussian noise with standard deviation σ_{Noise}. Then the total error will actually be given by

$$\text{Error} = \text{Accelerometer Output} - \text{Theory} = g\cos\theta_K^* + B + SFg\cos\theta_k^*$$
$$+ K(g\cos\theta_K^*)^2 - g\cos\theta_K$$

where θ_k is the actual angle and θ_k^* is the measured angle. The true values for the bias, scale-factor, and g-sensitive drift errors are repeated from Chapter 2 and are summarized in Table 4.3.

Table 4.3 Nominal values for accelerometer testing example

Term	Scientific value	English units
Bias error	10 μg	$10*10^{-6}*32.2 = 0.000322\ \mathrm{ft/s^2}$
Scale-factor error	5 ppm	$5*10^{-6}$
g-squared sensitive drift	1 μg/g^2	$1*10^{-6}/32.2 = 3.106*10^{-8}\mathrm{s^2/ft}$

To put the information of this section into a format suitable for Kalman filtering, we first have to choose a set of states for the filter to estimate. Because we would like to estimate the bias, scale-factor error, and g-squared sensitive drift, these would be obvious choices for states. If these states are constants, their derivatives must be zero and the state-space equation, upon which our Kalman filter will be designed, is given by

$$\begin{bmatrix} \dot{B} \\ \dot{SF} \\ \dot{K} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} B \\ SF \\ K \end{bmatrix}$$

We have neglected process noise in this formulation. Because the systems dynamics matrix can be seen from the preceding equation to be zero or

$$\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

the discrete fundamental matrix must be the identity matrix because

$$\Phi_k = \boldsymbol{I} + \boldsymbol{F}T_s + \cdots = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} T_s = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We can assume the measurement is simply the error already defined:

$$z_k = g\cos\theta_K^* + B + SFg\cos\theta_K^* + K(g\cos\theta_K^*)^2 - g\cos\theta_K$$

Therefore, the measurement is linearly related to the states if we have

$$z_k = [1 \quad g\cos\theta_K^* \quad (g\cos\theta_K^*)^2] \begin{bmatrix} B \\ SF \\ K \end{bmatrix} + v_k$$

where the measurement matrix can be seen from the preceding equation to be

$$H = [1 \quad g\cos\theta_K^* \quad (g\cos\theta_K^*)^2]$$

and v_k is the measurement noise. The measurement noise is considered to be the difference between the actual accelerometer reading and the theoretical accelerometer output or

$$v_k = g\cos\theta_K^* - g\cos\theta_K = g(\cos\theta_K^* - \cos\theta_K)$$

The actual noise is on the angle (i.e., θ_k^*), and our job is to find an equivalent noise v_k. The method for finding the equivalent noise is to first assume that the noisy angle is simply the true angle plus a small term. Therefore, the noise term v_k becomes

$$v_k = g[\cos(\theta_K + \Delta\theta_k) - \cos\theta_K]$$

Using the trigonometric expansion

$$\cos(\theta_K + \Delta\theta_k) = \cos\theta_k\cos\Delta\theta_k - \sin\theta_k\sin\Delta\theta_k$$

and making the small angle approximation, we obtain

$$v_k = g(\cos\theta_k\cos\Delta\theta_k - \sin\theta_k\sin\Delta\theta_k - \cos\theta_K) \approx -g\Delta\theta_k\sin\theta_k$$

Squaring and taking expectations of both sides of the preceding equation yields an expression for the variance of the effective measurement noise as

$$\boldsymbol{R}_k = E(v_k^2) = (g\sin\theta_k)^2 E(\Delta\theta_k^2) = g^2\sin^2\theta_k\sigma_\theta^2$$

In other words, we have just developed an expression for the variance of the equivalent noise or pseudonoise $\boldsymbol{R}_k$ in terms of the variance of the actual noise σ_θ^2. Assuming zero process noise, we now have enough information to build the Kalman filter and solve the Riccati equations for the Kalman gains. Because there is no deterministic input in this formulation of the problem, the Kalman-filtering equation simplifies to

$$\hat{\boldsymbol{x}}_k = \Phi_k\hat{x}_{k-1} + \boldsymbol{K}_k(z_k - \boldsymbol{H}\Phi_k\hat{\boldsymbol{x}}_{k-1})$$

Substitution of the appropriate matrices yields

$$\begin{bmatrix}\hat{B}_k\\ \widehat{SF}_k\\ \hat{K}_k\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\hat{B}_{k-1}\\ \widehat{SF}_{k-1}\\ \hat{K}_{k-1}\end{bmatrix}$$
$$+\begin{bmatrix}K_{1_k}\\ K_{2_k}\\ K_{3_k}\end{bmatrix}\left[z_k - [1 \quad g\cos\theta_K^* \quad (g\cos\theta_K^*)^2]\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\hat{B}_{k-1}\\ SF_{k-1}\\ \hat{K}_{k-1}\end{bmatrix}\right.$$

where the measurement z_k has already been defined. Multiplying out the preceding matrix difference equation and defining a residual yields the scalar equations

$$\begin{aligned}
\mathrm{Res}_k &= z_k - \widehat{BIAS}_{k-1} - \widehat{SF}_{k-1} g\cos\theta_k^* - \hat{K}_{k-1}(g\cos\theta_k^*)^2 \\
\hat{B}_k &= \hat{B}_{k-1} + K_{1_k}\mathrm{Res}_k \\
\widehat{SF}_k &= \widehat{SF}_{k-1} + K_{2_k}\mathrm{Res}_k \\
\hat{K}_k &= \hat{K}_{k-1} + K_{3_k}\mathrm{Res}_k
\end{aligned}$$

Listing 4.5 presents the simulation that uses the Kalman filter for estimating the accelerometer bias, scale-factor, and g-sensitive drift. We can see from Listing 4.5 that the filter states are initialized to zero (i.e., $BIASH = SFH = XKH = 0$) because we do not have any a priori information. The angle measurement noise is initially set to 1 μr. Other parameters have been set in accordance with Table 4.3. We print out the filter state estimates along with the true states. In addition, we compute the actual error in the state estimates along with the covariance matrix predictions. The simulation also checks to see if the actual single-run measurement noise

$$v_k = g\cos\theta_K^* - g\cos\theta_K$$

falls between the theoretically calculated plus and minus value of the standard deviation of the measurement noise

$$\sigma_{v_k} = g\sin\theta_k\sigma_\theta$$

The nominal case of Listing 4.5 was run, and the first check was to see if the single-run measurement noise was within the theoretical bounds of the measurement noise standard deviation which was derived in this section. We can see from Fig. 4.38 that there appears to be an agreement between the single-run simulation results and the derived formula (denoted "Theory" in the figure), which indicates that we derived the pseudonoise formula correctly.

The first test to see if the filter is working properly is to see if the actual errors in the estimates lie within the theoretical bounds as determined by the square root of the appropriate diagonal element of the covariance matrix. The nominal case of Listing 4.5 was run, and Figs. 4.39–4.41 present the errors in the estimates for all three states. We can see that because in all three cases the single flight results are within the theoretical bounds the filter appears to be working properly.

We would also like to see how well the actual state estimates compared to the true error terms. Because there is noise on the measurement angle, the state estimates will vary from run to run. Listing 4.5 was modified slightly so that it could be run in the Monte Carlo mode. Cases were run in which the angle measurement noise was 1 and 10 μr. We can see from Figs. 4.42–4.44 that when the angle measurement noise is 1 μr the filter is able to estimate each of the error terms quite well. Of course, for this case there was 1 μr of angle noise. We

Listing 4.5 Simulation of Kalman filter for estimating accelerometer bias, scale-factor, and *g*-sensitive drift

```
C THE FIRST THREE STATEMENTS INVOKE THE ABSOFT RANDOM
  NUMBER GENERATOR ON THE MACINTOSH
      GLOBAL DEFINE
            INCLUDE 'quickdraw.inc'
      END
      IMPLICIT REAL*8 (A-H)
      IMPLICIT REAL*8 (O-Z)
      REAL*8 PHI(3,3),P(3,3),M(3,3),PHIP(3,3),PHIPPHIT(3,3),K(3,1)
      REAL*8 Q(3,3),HMAT(1,3),HM(1,3),MHT(3,1)
      REAL*8 PHIT(3,3),R(1,1)
      REAL*8 HMHT(1,1),HT(3,1),KH(3,3),IDN(3,3),IKH(3,3)
      INTEGER ORDER
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      ORDER=3
      BIAS=.00001*32.2
      SF=.000005
      XK=.000001/32.2
      SIGTH=.000001
      G=32.2
      BIASH=0.
      SFH=0.
      XKH=0
      SIGNOISE=.000001
      S=0.
      DO 1000 I=1,ORDER
      DO 1000 J=1,ORDER
            PHI(I,J)=0.
            P(I,J)=0.
            Q(I,J)=0.
            IDN(I,J)=0.
1000  CONTINUE
      IDN(1,1)=1.
      IDN(2,2)=1.
      IDN(3,3)=1.
      PHI(1,1)=1.
      PHI(2,2)=1.
      PHI(3,3)=1.
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      P(1,1)=9999999999.
      P(2,2)=9999999999.
      P(3,3)=9999999999.
      DO 1100 I=1,ORDER
            HMAT(1,I)=0.
            HT(I,1)=0.
1100  CONTINUE
      DO 10 THETDEG=0.,180.,2.
         THET=THETDEG/57.3
         CALL GAUSS(THETNOISE,SIGTH)
```

(*continued*)

Listing 4.5 (*Continued*)

```
              THETS=THET+THETNOISE
              HMAT(1,1)=1
              HMAT(1,2)=G*COS(THETS)
              HMAT(1,3)=(G*COS(THETS))**2
              CALL MATTRN(HMAT,1,ORDER,HT)
              R(1,1)=(G*SIN(THETS)*SIGTH)**2
              CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
              CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,
                   PHIPPHIT)
              CALL MATADD(PHIPPHIT,ORDER,ORDER,Q,M)
              CALL MATMUL(HMAT,1,ORDER,M,ORDER,ORDER,HM)
              CALL MATMUL(HM,1,ORDER,HT,ORDER,1,HMHT)
              HMHTR=HMHT(1,1)+R(1,1)
              HMHTRINV=1./HMHTR
              CALL MATMUL(M,ORDER,ORDER,HT,ORDER,1,MHT)
              DO 150 I=1,ORDER
                     K(I,1)=MHT(I,1)*HMHTRINV
150           CONTINUE
              CALL MATMUL(K,ORDER,1,HMAT,1,ORDER,KH)
              CALL MATSUB(IDN,ORDER,ORDER,KH,IKH)
              CALL MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
              Z=BIAS+SF*G*COS(THETS)+XK*(G*COS(THETS))**2-
              G*COS(THET)+G*COS(THETS)
              RES=Z-BIASH-SFH*G*COS(THETS)-XKH*(G*COS(THETS))**2
              BIASH=BIASH+K(1,1)*RES
              SFH=SFH+K(2,1)*RES
              XKH=XKH+K(3,1)*RES
              SP11=SQRT(P(1,1))
              SP22=SQRT(P(2,2))
              SP33=SQRT(P(3,3))
              BIASERR=BIAS-BIASH
              SFERR=SF-SFH
              XKERR=XK-XKH
              ACTNOISE=G*COS(THETS)-G*COS(THET)
              SIGR=SQRT(R(1,1))
              WRITE(9,*)THETDEG,BIAS,BIASH,SF,SFH,XK,XKH
              WRITE(1,*)THETDEG,BIAS,BIASH,SF,SFH,XK,XKH
              WRITE(2,*)THETDEG,BIASERR,SP11,-SP11,SFERR,SP22,-SP22,
1               XKERR,SP33,-SP33,ACTNOISE,SIGR,-SIGR
10      CONTINUE
        CLOSE(1)
        PAUSE
        END

C SUBROUTINE GAUSS IS SHOWN IN LISTING 1.8
C SUBROUTINE MATTRN IS SHOWN IN LISTING 1.3
C SUBROUTINE MATMUL IS SHOWN IN LISTING 1.4
C SUBROUTINE MATADD IS SHOWN IN LISTING 1.1
C SUBROUTINE MATSUB IS SHOWN IN LISTING 1.2
```

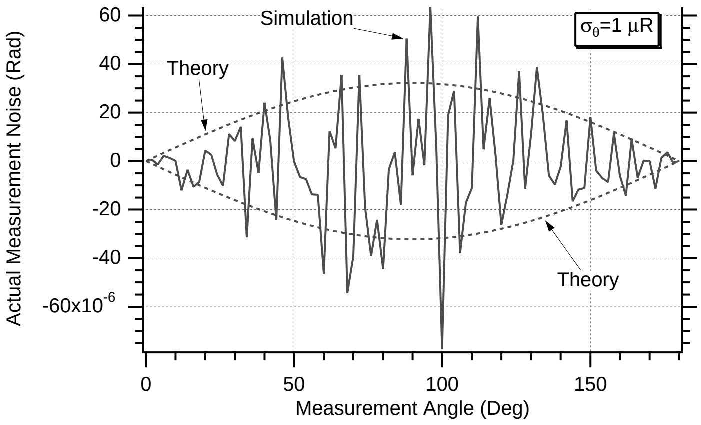

Fig. 4.38 Derived formula for standard deviation of pseudonoise appears to be correct.

showed in Chapter 2 that the method of least squares also worked quite well. However, we can also see from the three figures that when the measurement angle noise is increased to 10 μr all the estimates deteriorate. In fact for 10 μr of angle measurement noise, it is now impossible to determine the *g*-sensitive drift.

In Chapter 2 we saw that when using the method of least squares we were not able to estimate bias, scale-factor, and *g*-sensitive drift if there were 100 μr of measurement noise. To see if things get better when a Kalman filter is used, we now rerun Listing 4.5 with 100 μr of measurement noise. We can see from Figs. 4.45–4.47 that the Kalman filter also cannot estimate bias, scale-factor, and *g*-sensitive drift when there are 100 μr of measurement noise. Thus, nothing magical happens when a Kalman filter is used.

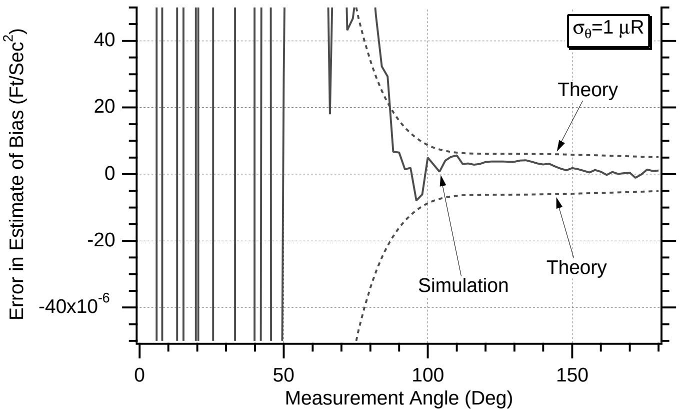

Fig. 4.39 Kalman filter appears to be working correctly because actual error in estimate of accelerometer bias is within theoretical bounds.

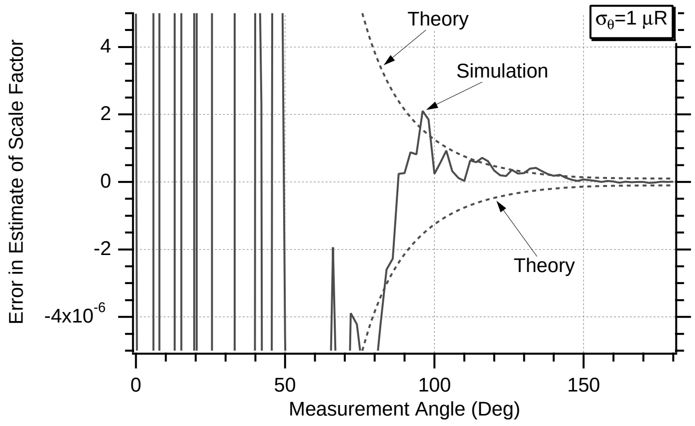

Fig. 4.40 Kalman filter appears to be working correctly because actual error in estimate of accelerometer scale-factor error is within theoretical bounds.

Summary

In this chapter we presented the theoretical equations for making a discrete Kalman filter. We showed, via several simulation examples, that a polynomial Kalman filter with zero process noise and infinite covariance matrix (i.e., diagonal elements are infinite and off-diagonal elements are zero) was equivalent to the recursive least-squares filters of the preceding chapter. We also showed that the performance of the polynomial Kalman filter was approximately independent of the initial covariance matrix for covariances above relatively small values. A numerical example was presented illustrating various Kalman-filtering options for the problem of tracking a falling object. We showed that when the Kalman filter

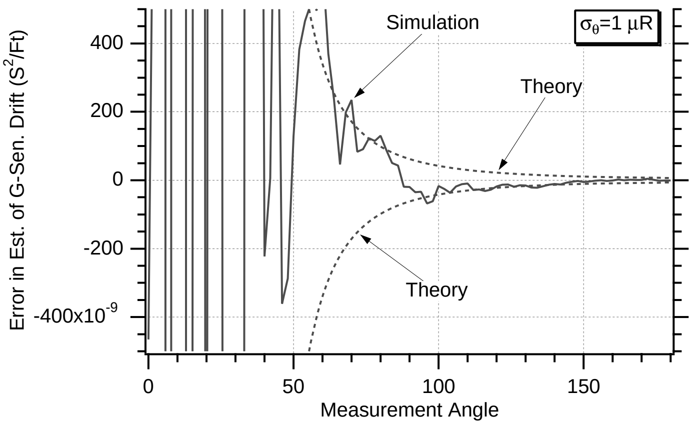

Fig. 4.41 Kalman filter appears to be working correctly because actual error in estimate of accelerometer *g*-sensitive drift is within theoretical bounds.

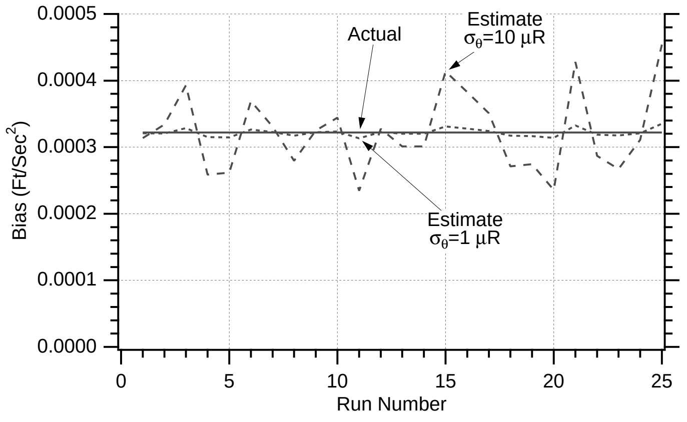

Fig. 4.42 Kalman filter estimates accelerometer bias accurately.

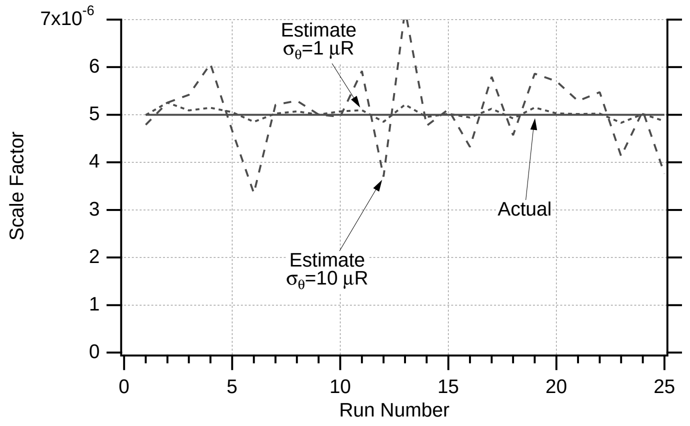

Fig. 4.43 Kalman filter estimates accelerometer scale-factor error accurately.

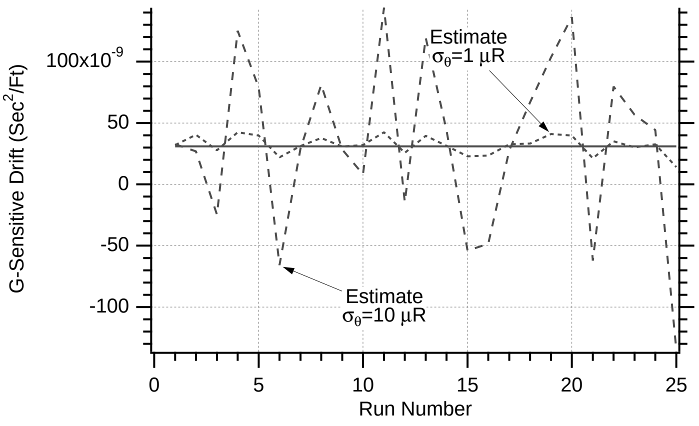

Fig. 4.44 Kalman filter estimates accelerometer *g*-sensitive drift accurately.

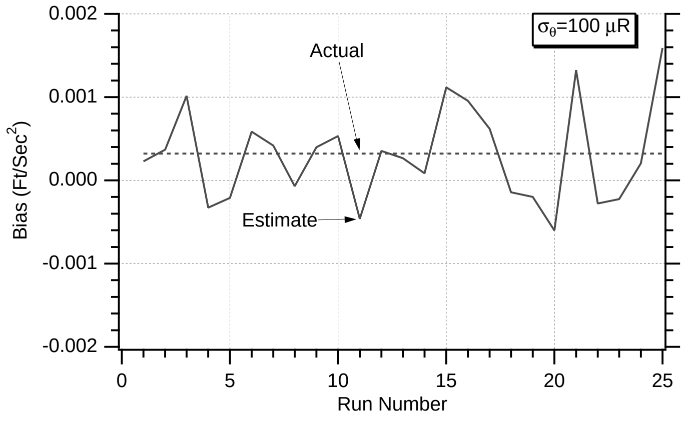

Fig. 4.45 Kalman filter is not able to estimate accelerometer bias when there are 100 μr of measurement noise.

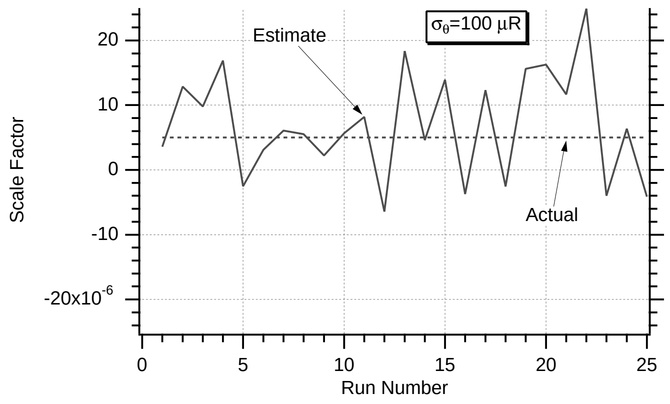

Fig. 4.46 Kalman filter is not able to estimate accelerometer scale-factor error when there are 100 μr of measurement noise.

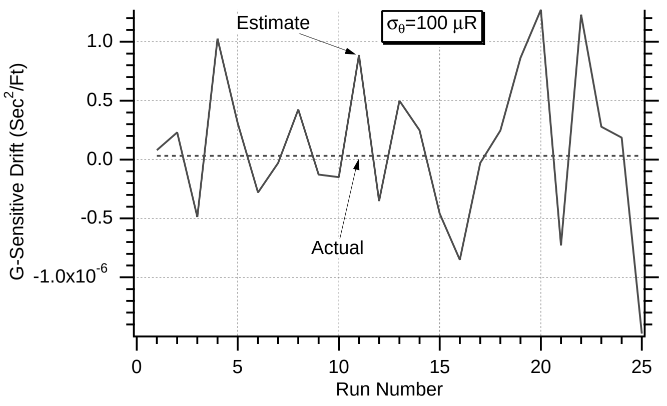

Fig. 4.47 Kalman filter is not able to estimate accelerometer *g*-sensitive drift when there are 100 μr of measurement noise.

was able to make use of a priori information superior performance could be obtained. We also showed that the addition of process noise could sometimes be used as a tool for preventing filter divergence.

References

[1]Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, Vol. 82, No. 1, 1960, pp. 35–46.

[2]Gelb., A., *Applied Optimal Estimation*, Massachusetts Inst. of Technology, Cambridge, MA 1974, pp. 102–155.

[3]Selby, S. M., *Standard Mathematical Tables*, 20th ed. Chemical Rubber Co., Cleveland, OH, 1972, pp. 491–499.