Chapter 5

Kalman Filters in a Nonpolynomial World

Introduction

SO FAR we have seen how polynomial Kalman filters perform when the measurement is a polynomial signal plus noise. In the real world most measurements cannot be described exactly by simple polynomials, and it is therefore of considerable practical interest to see how well the polynomial Kalman filter performs under these circumstances and if anything can be done to improve performance. Two examples will be picked in which the real world cannot be described by a polynomial signal corrupted by noise. Several possible Kalman-filter designs will be presented to illustrate key concepts.

Polynomial Kalman Filter and Sinusoidal Measurement

Suppose that the actual measurement is a pure sine wave of unity amplitude corrupted by noise or

$$x^* = \sin \omega t + \text{noise}$$

where ω is the frequency of the sinusoidal signal and the noise is zero-mean Gaussian with a standard deviation of unity. We would like to build a Kalman filter that will be able to track the sinusoid and estimate its states based on the noisy sinusoidal measurement.

Because the true signal is the sinusoid

$$x = \sin \omega t$$

its derivative is given by

$$\dot{x} = \omega \cos \omega t$$

For the sinusoidal measurement signal we can first attempt to use the first-order polynomial Kalman filter. The polynomial Kalman filter is a reasonable choice because at this point we have not yet studied any other kind of Kalman filter.

Recall that the general formula for the Kalman filter, assuming no known deterministic inputs, is given by

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

For the first-order polynomial Kalman filter we have already shown that the fundamental and measurement matrices are given by

$$\Phi_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$$
$$\boldsymbol{H} = [1 \quad 0]$$

Therefore, after some algebra the scalar equations representing the first-order polynomial Kalman filter can be shown to be

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

where the residual is defined as

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

We have shown already that the Kalman gains $\boldsymbol{K}_k$, required by the preceding set of discrete filtering equations, are obtained from the following recursive set of discrete matrix Riccati equations:

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

In this case the matrix describing the variance of the measurement noise turns out to be a scalar given by

$$\boldsymbol{R}_K = \sigma_n^2$$

whereas the matrix describing the process noise has already been shown in Chapter 4 to be

$$\boldsymbol{Q}_k = \Phi_s \begin{bmatrix} \dfrac{T_s^3}{3} & \dfrac{T_s^2}{2} \\ \dfrac{T_s^2}{2} & T_s \end{bmatrix}$$

Listing 5.1 presents a first-order polynomial Kalman filter measuring the noisy sinusoidal signal and attempting to estimate the first two states of the sinusoid

(i.e., X and XD). In the nominal case shown in Listing 5.1, the frequency of the sinusoid is 1 rad/s while its amplitude is unity. As was already mentioned, for the nominal example the measurement noise is Gaussian with zero mean and unity standard deviation. A measurement is being taken every 0.1 s (i.e TS = 0.1). Initially the Kalman filter is set up without process noise (PHIS = 0), and the initial state estimates are set to zero. As before, the diagonal terms of the initial covariance matrix are set to infinity to indicate that we have no idea how to initialize the filter. Printed out every sampling time are the true signal X and its estimate XH and the true derivative XD and its estimate XDH.

Listing 5.1 First-order polynomial Kalman filter and sinusoidal measurement

```
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 ORDER
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      ORDER=2
      PHIS=0.
      TS=.1
      XH=0.
      XDH=0.
      SIGNOISE=1.
      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(2,2)=99999999999.
      PHI(1,1)=1
      PHI(1,2)=TS
      PHI(2,2)=1
      HMAT(1,1)=1.
      HMAT(1,2)=0.
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      CALL MATTRN(HMAT,1,ORDER,HT)
```

(*continued*)

Listing 5.1 (*Continued*)

```
      Q(1,1)=PHIS*TS**3/3
      Q(1,2)=PHIS*TS*TS/2
      Q(2,1)=Q(1,2)
      Q(2,2)=PHIS*TS
      DO 10 T=0.,20.,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(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=SIN(T)
            XD=COS(T)
            XS=X+XNOISE
            RES=XS-XH-TS*XDH
            XH=XH+XDH*TS+K(1,1)*RES
            XDH=XDH+K(2,1)*RES
            WRITE(9,*)T,X,XS,XH,XD,XDH
            WRITE(1,*)T,X,XS,XH,XD,XDH
10    CONTINUE
      PAUSE
      CLOSE(1)
      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
```

Figure 5.1 shows the true signal along with the measurement as a function of time. We can see that the measurement appears to be very noisy for this example. From this noisy measurement the first-order polynomial Kalman filter will attempt to estimate the true signal and its derivative.

Figure 5.2 shows that even though the true signal is not a first-order polynomial the polynomial Kalman filter is not doing a bad job in extracting the true signal from the noisy measurement. However, in all honesty it would be difficult to use the filter's estimate of the first state to determine with certainty that the true signal was a sinusoid.

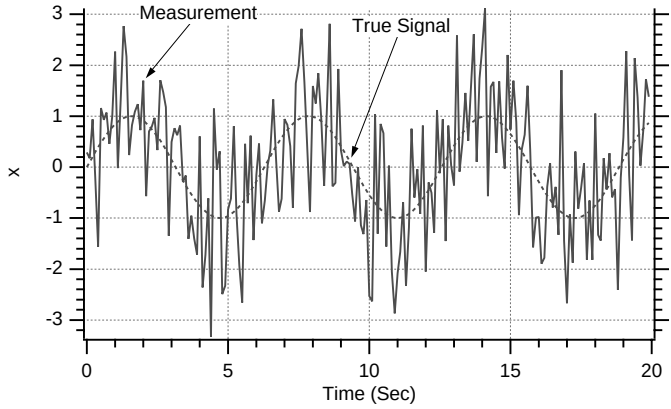

Fig. 5.1 Sinusoidal measurement is very noisy.

Figure 5.3 shows that matters get worse when the filter attempts to estimate the derivative to the actual signal, x dot. There is a large transient at the beginning because the filter has been initialized incorrectly (i.e., second filter state initialized to zero rather than unity). Although the filter transient settles out quickly, there is no sinusoidal motion in the estimate. Therefore, we can conclude that the first-order polynomial Kalman filter is doing a poor job of estimating the derivative of the true signal.

In a sense we have a modeling error when we are attempting to track a sinusoidal signal with a polynomial Kalman filter. We saw in Chapter 4 that one way of handling modeling errors was to increase the process noise. Figures 5.4 and 5.5 show that if we increase the process noise from zero to 10 (i.e., $\Phi_s = 0$ to $\Phi_s = 10$) matters improve. For example, Fig. 5.4 shows that the filter estimate of

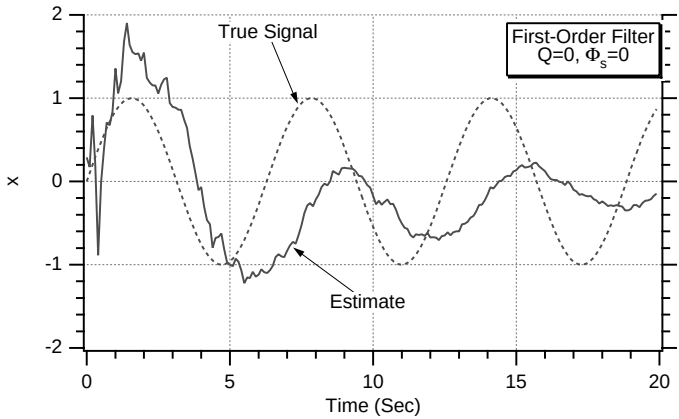

Fig. 5.2 First-order polynomial Kalman-filter has difficulty in tracking the sinusoidal signal.

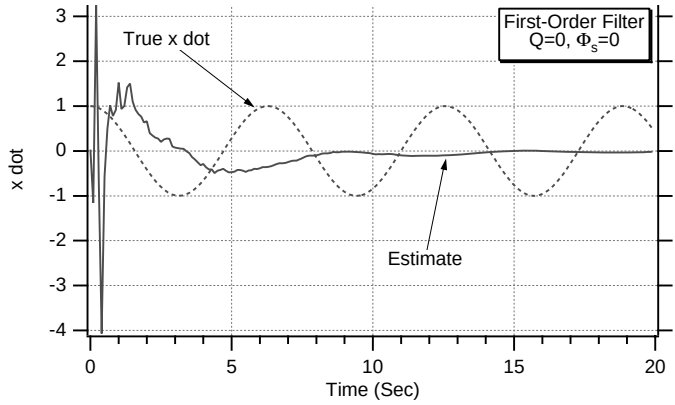

Fig. 5.3 First-order polynomial Kalman-filter does poorly in estimating the derivative of the true signal.

the first state now looks more sinusoidal. The price paid for being better able to track the signal is that the estimate is also noisier. Increasing the process noise effectively increases the bandwidth of the filter, which improves its tracking capabilities at the expense of more noise transmission. Figure 5.5 shows that the filter is now able to provide a noisy track of the derivative of the true signal. However, it is still difficult to discern the sinusoidal nature of this state from the noisy estimate.

It may be possible to improve tracking performance by using a higher-order polynomial Kalman filter. The higher-order polynomial should be a better match to the true sinusoidal signal. The reason for this is that one can show via a Taylor-series expansion that a sinusoid is really an infinite-order polynomial (see Chapter 1). Therefore, although we cannot use an infinite-order Kalman filter, we can try

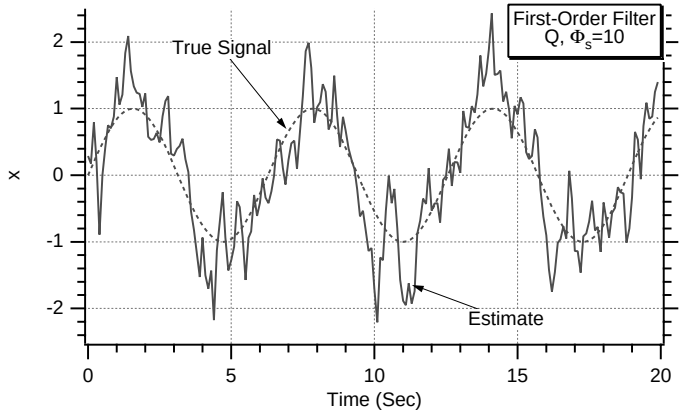

Fig. 5.4 Adding process noise yields better tracking at expense of noisier estimate.

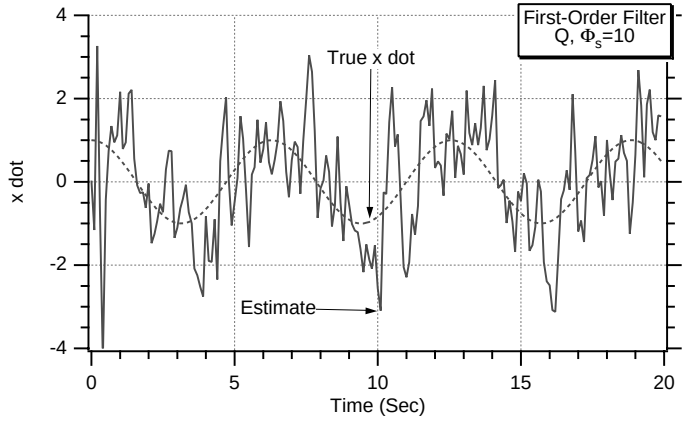

Fig. 5.5 First-order filter with process noise is now able to provide noisy estimate of derivative of true signal.

to use the second-order polynomial Kalman filter, which should be an improvement over the first-order Kalman filter. Again, recall that the general formula for the Kalman filter without any deterministic disturbances is given by

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

For the second-order polynomial Kalman filter we have already shown in Chapter 4 that the fundamental and measurement matrices are given by

$$\Phi_k = \begin{bmatrix} 1 & T_s & 0.5T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{H} = [1 \quad 0 \quad 0]$$

Therefore, after substitution and some algebra, the scalar equations representing the second-order filter can be shown to be

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

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

$$\hat{\ddot{x}}_k = \hat{\ddot{x}}_{k-1} + K_{3_k} \mathrm{Res}_k$$

where the residual is defined as

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

The Kalman gains $\boldsymbol{K}_k$, required by the filter, are still obtained from the recursive matrix Riccati equations

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

As was also true for the first-order filter, the matrix describing the variance of the measurement noise for the second-order filter turns out to be a scalar and is given by

$$\boldsymbol{R}_k = \sigma_n^2$$

whereas the matrix describing the process noise has already been shown in Chapter 4 to be

$$\boldsymbol{Q}_k = \Phi_s \begin{bmatrix} \frac{T_s^5}{20} & \frac{T_s^4}{8} & \frac{T_s^3}{6} \\ \frac{T_s^4}{8} & \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^3}{6} & \frac{T_s^2}{2} & T_s \end{bmatrix}$$

Listing 5.2 presents the simulation of the second-order polynomial Kalman filter measuring the noisy sinusoidal signal. As was also the case for the first-order filter, the frequency of the sinusoid is again 1 rad/s while the amplitude is unity. The Gaussian measurement noise has zero mean and unity standard deviation. Again, initially the Kalman filter is set up without process noise, and the initial state estimates are set to zero because of our lack of knowledge of

Listing 5.2 Second-order polynomial Kalman filter and sinusoidal measurement

```
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(3,3),Q(3,3),M(3,3),PHI(3,3),HMAT(1,3),HT(3,1),PHIT(3,3)
    REAL*8 RMAT(1,1),IDN(3,3),PHIP(3,3),PHIPPHIT(3,3),HM(1,3)
    REAL*8 HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),MHT(3,1),K(3,1)
    REAL*8 KH(23,3),IKH(3,3)
    INTEGER ORDER
```

(*continued*)

Listing 5.2 (*Continued*)

```
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      ORDER  =3
      PHIS=0.
      TS=.1
      XH=0.
      XDH=0.
      XDDH=0
      SIGNOISE=1.
      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.
      IDN(3,3)=1.
      P(1,1)=99999999999.
      P(2,2)=99999999999.
      P(3,3)=99999999999.
      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
      HMAT(1,1)=1.
      HMAT(1,2)=0.
      HMAT(1,3)=0.
      CALL  MATTRN(PHI,ORDER,ORDER,PHIT)
      CALL  MATTRN(HMAT,1,ORDER,HT)
      Q(1,1)=PHIS*TS**5/20
      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.,20.,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(HMAT,1,ORDER,M,ORDER,ORDER,HM)
              CALL  MATMUL(HM,1,ORDER,HT,ORDER,1,HMHT)
```

(*continued*)

Listing 5.2 (*Continued*)

```
            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=SIN(T)
            XD=COS(T)
            XDD=-SIN(T)
            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
            WRITE(9,*)T,X,XH,XD,XDH,XDD,XDDH
            WRITE(1,*)T,X,XH,XD,XDH,XDD,XDDH
10  CONTINUE
    PAUSE
    CLOSE(1)
    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
```

the real world. As before, the diagonal terms of the initial covariance matrix are set to infinity to reflect the fact that we have no a priori information.

Figure 5.6 shows that going to a higher-order polynomial Kalman filter with zero process noise definitely improves the filter's ability to track the actual signal. By comparing Fig. 5.6 with Fig. 5.2, we can see that the price paid for the better tracking ability is a somewhat noisier estimate. This is to be expected because we have already shown that higher-order filters have more noise transmission than lower-order filters. The fact that the process noise is zero makes this filter sluggish (i.e., lower bandwidth), which causes the filter estimate to lag the actual signal.

We can see from Fig. 5.7 that the second-order filter still has trouble in estimating the derivative of the true signal. Although the estimate is better than that of Fig. 5.3, it is still difficult to discern the sinusoidal nature of the derivative.

We saw in the case of the first-order polynomial Kalman filter that the estimates could be improved somewhat by adding process noise. Process noise was also added to the second-order polynomial Kalman filter. Figure 5.8 shows that the lag in the case in which there was no process noise (i.e., $\Phi_s = 0$) can be

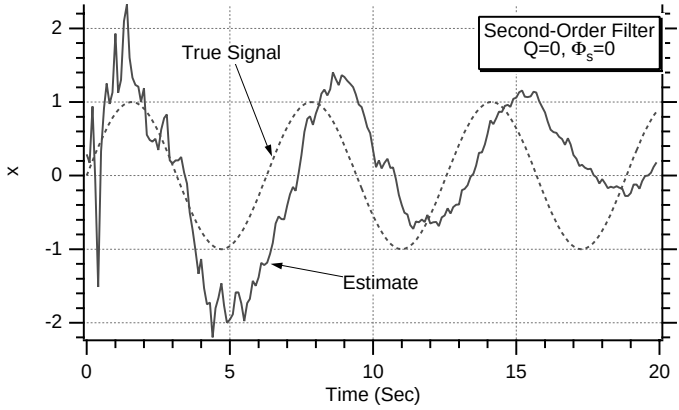

Fig. 5.6 Higher-order polynomial Kalman filter with zero process noise yields better but noisier estimates.

removed by adding process noise (i.e., $\Phi_s = 10$). We can see that the estimate of Fig. 5.8 is much better than that of Fig. 5.6. Figure 5.9 shows that adding process noise also improves the estimate of the derivative of the signal. The improvement in the estimate can be seen by comparing Fig. 5.9, in which there was process noise (i.e., $\Phi_s = 10$), with Fig. 5.7, in which there was no process noise (i.e., $\Phi_s = 0$). For the first time we are able to discern the sinusoidal nature of the derivative of the signal. In all of the preceding examples, we could have also handled the measurement $A \sin \omega t$, where A is the amplitude of the sinusoid and is different from unity. In this case the polynomial filters would not have any difficulty, and the estimates would simply be scaled by A.

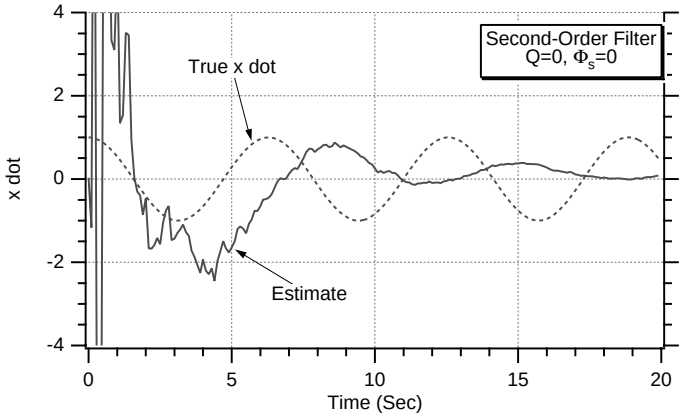

Fig. 5.7 Higher-order polynomial Kalman filter does better job of tracking derivative of true signal.

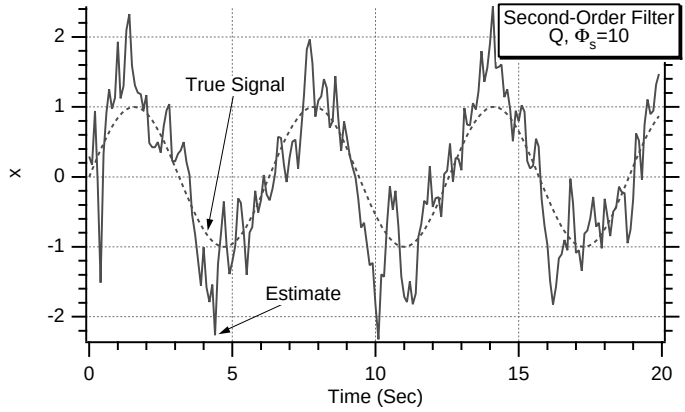

Fig. 5.8 Filter lag has been removed by the addition of process noise.

Sinusoidal Kalman Filter and Sinusoidal Measurement

So far we have seen how a second-order polynomial Kalman filter with process noise could track a sinusoidal signal. Its estimate of the derivative of the signal was not terrific but possibly acceptable for certain applications. The virtue of the polynomial Kalman filter was that no a priori information concerning the true signal was required. Could we have done better with another type of Kalman filter had we known that the true signal was sinusoidal? Let us see if we can improve filter performance if a priori information is available. Recall that the actual signal is

$$x = A \sin \omega t$$

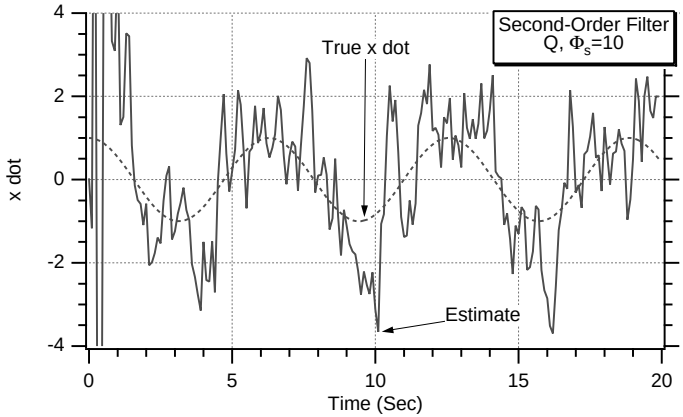

Fig. 5.9 Estimate of derivative has been improved by the addition of process noise.

If we take the derivative of this signal, we also get a sinusoid or

$$\dot{x} = A\omega \cos \omega t$$

Taking the derivative again yields

$$\ddot{x} = -A\omega^2 \sin \omega t$$

The preceding equation can be expressed in terms of the first equation. Therefore, we can rewrite the preceding equation as

$$\ddot{x} = -\omega^2 x$$

The sinusoidal term has been eliminated, and the second derivative of the signal or state has been expressed in terms of a state. The preceding differential equation does not depend on the amplitude of the sinusoid but only its frequency. We can rewrite the preceding equation as a matrix differential equation in state-space form or

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix}$$

This equation is our new model of the real world. Remember from Chapter 1 that the state-space form is just another way of expressing the scalar second-order differential equation. Although the state-space form adds no additional information, it is required for the work that follows. From the preceding equation we can see that the system dynamics matrix is

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

For a time-invariant systems dynamics matrix we can derive the fundamental matrix according to[1]

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

where $\boldsymbol{I}$ is the identity matrix. First, we express the inverse of $\boldsymbol{sI} - \boldsymbol{F}$ as

$$(\boldsymbol{sI} - \boldsymbol{F})^{-1} = \begin{bmatrix} s & -1 \\ \omega^2 & s \end{bmatrix}^{-1}$$

Taking the inverse of a 2×2 matrix can nearly be done by inspection or following the formula from Chapter 1 and can easily be shown to be

$$\Phi(\boldsymbol{s}) = (\boldsymbol{sI} - \boldsymbol{F})^{-1} = \frac{1}{s^2 + \omega^2} \begin{bmatrix} s & 1 \\ -\omega^2 & 1 \end{bmatrix}$$

We now have the fundamental matrix in the Laplace transform or s domain. We must take the inverse Laplace transform to express the fundamental matrix as a function of time. Using inverse Laplace transform tables[2] yields

$$\Phi(t) = \begin{bmatrix} \cos\omega t & \dfrac{\sin\omega t}{\omega} \\ -\omega\sin\omega t & \cos\omega t \end{bmatrix}$$

Thus, we can see that the fundamental matrix is now sinusoidal. To derive the discrete fundamental matrix, required for the Kalman filter and Riccati equations, we simply substitute the sampling time T_s for time or

$$\Phi_k = \begin{bmatrix} \cos\omega T_s & \dfrac{\sin\omega T_s}{\omega} \\ -\omega\sin\omega T_s & \cos\omega T_s \end{bmatrix}$$

Again, it is important to note that we can propagate the states forward exactly with the fundamental matrix exactly as long as we know the frequency of the sinusoid. We do not have to know the amplitude of the sinusoid. Substituting the new fundamental matrix 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 a matrix difference equation, which can easily be converted to several scalar difference equations or

$$\hat{x}_k = \cos\omega T_s\hat{x}_{k-1} + \frac{\sin\omega T_s}{\omega}\hat{\dot{x}}_{k-1} + K_{1_k}\mathrm{Res}_k$$
$$\hat{\dot{x}}_k = -\omega\sin\omega T_s\hat{x}_{k-1} + \cos\omega T_s\hat{\dot{x}}_{k-1} + K_{2_k}\mathrm{Res}_k$$

where the residual is defined as

$$RES_k = x_k^* - \cos\omega T_s\hat{x}_{k-1} - \frac{\sin\omega T_s}{\omega}\hat{\dot{x}}_{k-1}$$

In a similar way the Kalman gains can also be obtained using the new fundamental matrix. Remember that this Kalman filter assumes that we know the frequency of the sinusoid. If the frequency of the sinusoid is unknown and must also be estimated, nonlinear filtering techniques must be used (see Chapter 10, which uses an extended Kalman filter to estimate the same states plus the frequency of the sinusoid).

Listing 5.3 shows the new two-state or first-order Kalman filter that makes use of the fact that the signal is sinusoidal. In the nominal case the frequency of the sinusoid is again 1 rad/s, while the amplitude is unity (i.e., running with different amplitudes will not change the results qualitatively). The Gaussian measurement noise again has zero mean and unity standard deviation. Initially the Kalman filter

Listing 5.3 New first-order Kalman filter and sinusoidal measurement

```
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 ORDER
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      ORDER=2
      PHIS=0.
      W=1
      A=1
      TS=.1
      XH=0.
      XDH=0.
      SIGNOISE=1.
      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(2,2)=99999999999.
      PHI(1,1)=COS(W*TS)
      PHI(1,2)=SIN(W*TS)/W
      PHI(2,1)=-W*SIN(W*TS)
      PHI(2,2)=COS(W*TS)
      HMAT(1,1)=1.
      HMAT(1,2)=0.
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      CALL MATTRN(HMAT,1,ORDER,HT)
      Q(1,1)=PHIS*TS**3/3
      Q(1,2)=PHIS*TS*TS/2
      Q(2,1)=Q(1,2)
      Q(2,2)=PHIS*TS
      DO 10 T=0.,20.,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)
```

(*continued*)

Listing 5.3 (*Continued*)

```
            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=A*SIN(W*T)
            XD=A*W*COS(W*T)
            XS=X+XNOISE
            XHOLD=XH
            RES=XS-XH*COS(W*TS)-SIN(W*TS)*XDH/W
            XH=COS(W*TS)*XH+XDH*SIN(W*TS)/W+K(1,1)*RES
            XDH=-W*SIN(W*TS)*XHOLD+XDH*COS(W*TS)+K(2,1)*RES
            WRITE(9,*)T,X,XS,XH,XD,XDH
            WRITE(1,*)T,X,XS,XH,XD,XDH
 10 CONTINUE
    PAUSE
    CLOSE(1)
    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
```

is set up without process noise, and the initial state estimates are set to zero. As before, the diagonal terms of the initial covariance matrix are set to infinity.

The nominal case of Listing 5.3 was run, and Fig. 5.10 shows that the new Kalman filter, which makes use of the fact that the signal is sinusoidal, does a spectacular job in estimating the signal when there is no process noise. The process noise was set to zero because we knew there were no modeling errors. We can see that not having process noise reduced the noise transmission. Note also from Fig. 5.10 that there is virtually no lag between the actual signal and the estimate. We can also see from Fig. 5.11 that the new Kalman filter's estimate of the derivative of the signal is near perfect. Thus, we can see that if a priori information is really available (i.e., actual signal is a sinusoid) it pays to use it. Of course if the real signal were really a polynomial and we thought it was supposed to be a sinusoid, a serious error could result.

In practice it is often too difficult to find the fundamental matrix exactly if we are using the Taylor-series approach. We have already derived the exact fundamental matrix using the Taylor-series approach for a sinusoidal signal in Chapter 1. We shall now investigate the influence on system performance when an

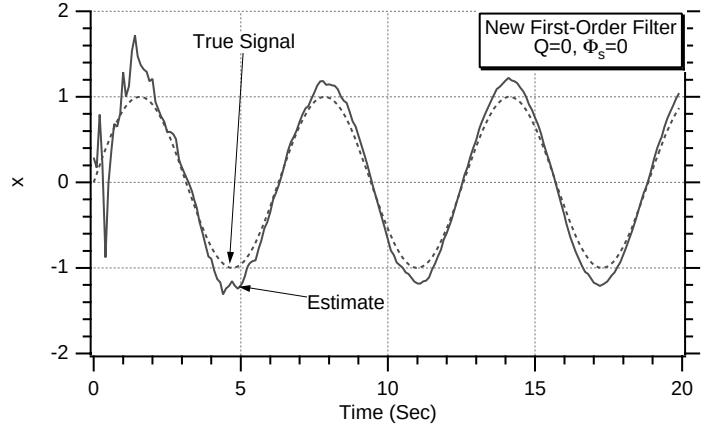

Fig. 5.10 New filter dramatically improves estimate of signal.

approximate fundamental matrix is used instead. The fundamental matrix can be expressed in terms of the systems dynamics matrix as

$$\Phi_k = e^{FT_s} \approx I + FT_s + \frac{(FT_s)^2}{2} + \cdots$$

If we approximate the preceding expression with a two-term Taylor series for the fundamental matrix, we obtain

$$\Phi_k \approx I + FT_s = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} T_s$$

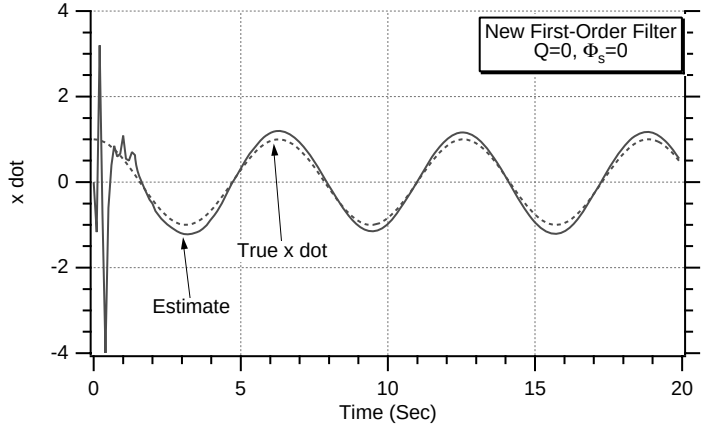

Fig. 5.11 New filter dramatically improves estimate of derivative of signal.

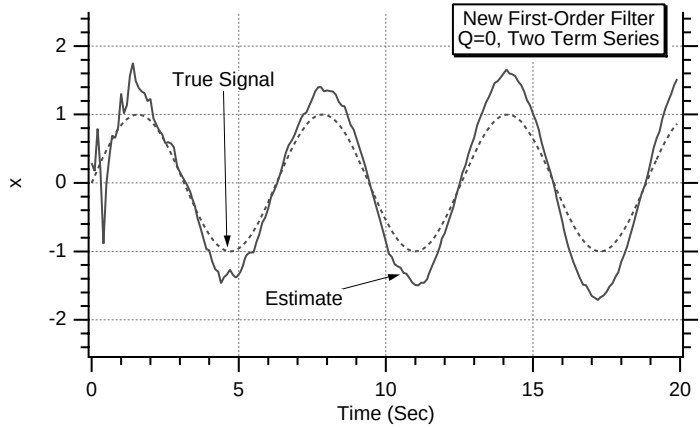

Fig. 5.12 Estimate of signal is worse when fundamental matrix is approximate.

or, more compactly,

$$\Phi_k \approx \begin{bmatrix} 1 & T_s \\ -\omega^2 T_s & 1 \end{bmatrix}$$

The new Kalman filter was rerun, but using the approximate fundamental matrix rather than the exact one. We can see from Figs. 5.12 and 5.13 that although the estimates of the signal and its derivative still appear to be sinusoidal in shape they are not as good as the estimates based on the exact fundamental matrix in Figs. 5.10 and 5.11.

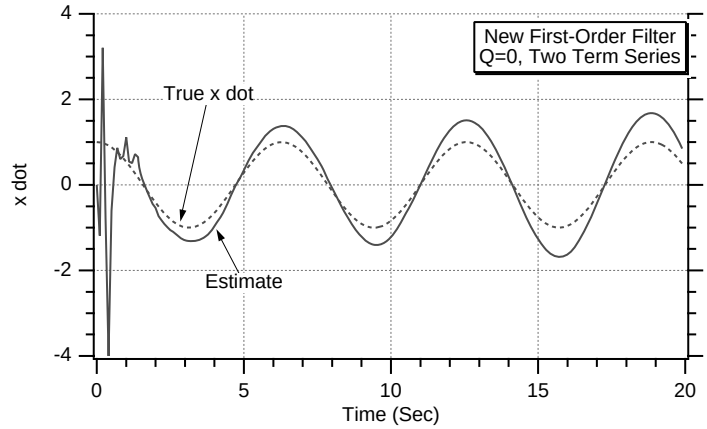

Fig. 5.13 Estimate of signal derivative is also worse when fundamental matrix is approximate.

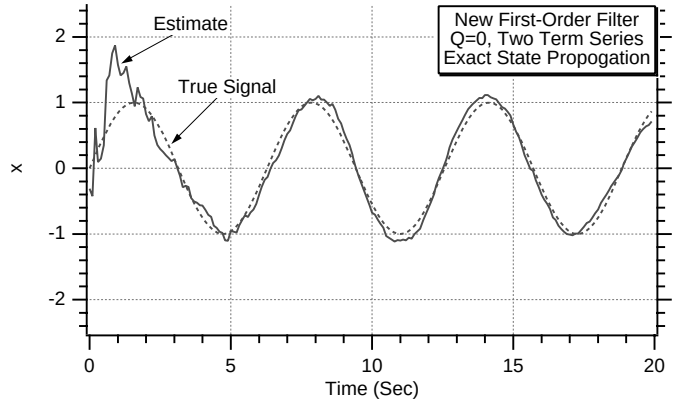

Fig. 5.14 Estimate of signal is excellent when two-term Taylor-series approximation for fundamental matrix is only used in Riccati equation.

Let us now assume that the approximate fundamental matrix, based on the two-term Taylor-series expansion, was used only in the Riccati equations to compute the Kalman gains, and the exact fundamental matrix was used to propagate the states in the filter. This example represents an important case, which we shall examine in more detail when we discuss extended Kalman filters. With extended Kalman filters the filtering equations are propagated forward by actual numerical integration of the state equations (i.e., an exact fundamental matrix in this case represents a perfect integration), and the fundamental matrix (i.e., often approximate because it is based on a few terms from a Taylor-series expansion) is only used in the Riccati equations. We can see from Figs. 5.14 and 5.15 that now the performance of the Kalman filter is excellent when the states

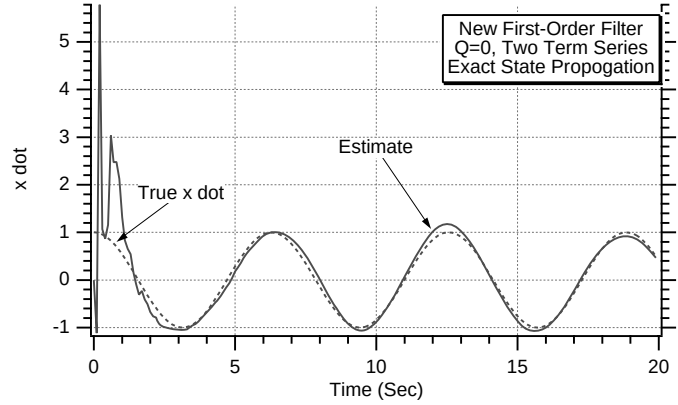

Fig. 5.15 Estimate of signal derivative is also excellent when two-term Taylor-series approximation for fundamental matrix is only used in Riccati equation.

are propagated correctly even though we are using suboptimal Kalman gains. This leads us to believe that the filter structure is more important than the gain computation.

Let us now assume that the fundamental matrix was known precisely. However, our knowledge of the signal frequency was in error by 100%. The real signal frequency was 1 rad/s, but in the fundamental matrix and in the filter we thought the weave frequency was actually 2 rad/s. Figure 5.16 shows that the estimate of the signal deteriorates substantially. This shows the sensitivity of this type of Kalman filter to modeling errors. The polynomial Kalman filter would have been insensitive to this type of error because it made no use of knowledge of the frequency ω of the sinusoidal signal.

To further demonstrate that the deterioration in performance caused by the incorrect fundamental matrix was because of an error in filter structure, rather than an error in filter gains, another experiment was conducted based on the results of Fig. 5.16. The preceding example was repeated, except this time the frequency mismatch only applied to the Riccati equation fundamental matrix. The actual fundamental matrix was used to propagate the states in the Kalman filter. We can see from Fig. 5.17 that now the filter estimates are excellent. This experiment again demonstrates that the filter structure is more important than the gain computation.

Often there really is a mismatch between the real world and the filtering world—both in the filter structure and in the gain computation. Under these circumstances the engineering fix to the mismatched frequency situation of Fig. 5.16 is to add process noise to the Kalman filter. The addition of process noise tells the filter that our model of the real world may be in error. The more process noise we add, the less reliable our model of the real world. Figure 5.18 shows that if we added process noise by making $\Phi_s = 10$ rather than $\Phi_s = 0$, then the estimate of the true signal improves substantially. However, the quality of the estimate is approximately similar to that provided by an ordinary polynomial Kalman filter (see Fig. 5.8). We will revisit this problem in Chapter 10.

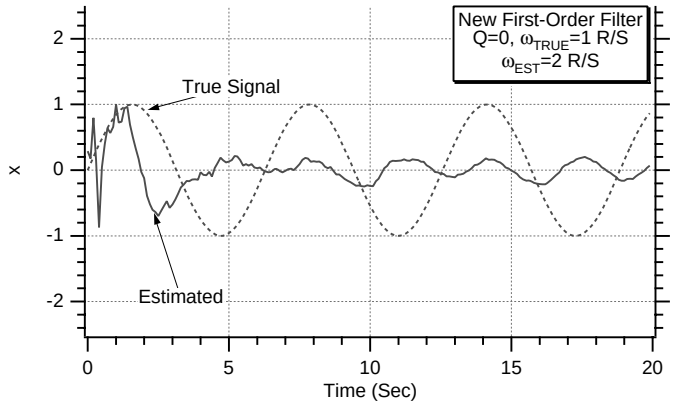

Fig. 5.16 Kalman filter that depends on knowing signal frequency is sensitive to modeling errors.

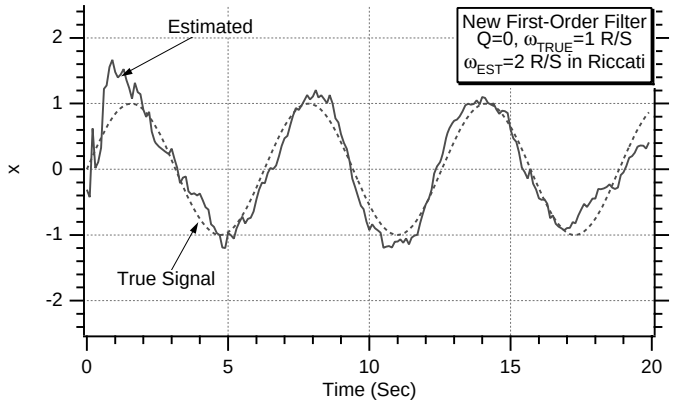

Fig. 5.17 If frequency mismatch is in Riccati equations only, filter still gives excellent estimates.

Suspension System Example

Consider the case of a car riding over a bumpy road. A suspension system that is attached to the wheel axle and car body frame protects the car from the bumps in the road. Figure 5.19 simplifies the example considerably by assuming that the car has only one wheel. For analytical convenience the bumpy road is represented by the sinusoid

$$x_2 = A_2 \sin \omega t$$

The wheel has radius b_1, and the length of the suspension is denoted $x_1 + b_2$. As one would expect, the purpose of the suspension is to provide the car with a

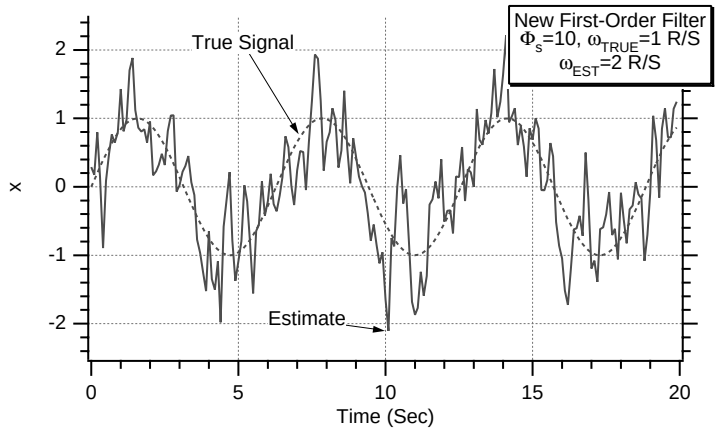

Fig. 5.18 Adding process noise enables Kalman filter with bad a priori information to provide good estimates.

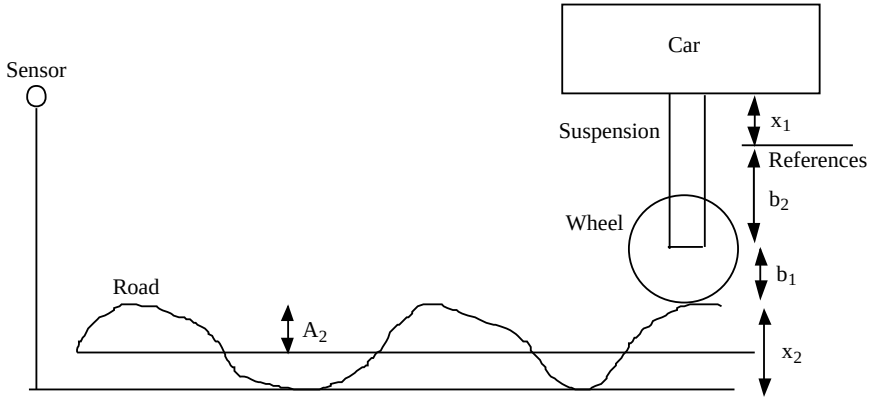

Fig. 5.19 Car riding over a bumpy road.

smooth ride. Therefore, if the suspension system is working properly, the variations in the height of the car $x_1 + b_2 + b_1 + x_2$ will be much smaller than that of the road amplitude A_2. A sensor measures the height of the car at all times so we can see if the suspension is working properly.

The suspension system of Fig. 5.19 is drawn in more detail in Fig. 5.20. For analytical convenience we have represented the suspension system by a spring and a dashpot.[3,4] The stiffness of the spring is denoted by its spring constant K, which by definition is the number of pounds tension necessary to extend the spring 1 in. In this example the car is considered to have mass M. The dashpot of

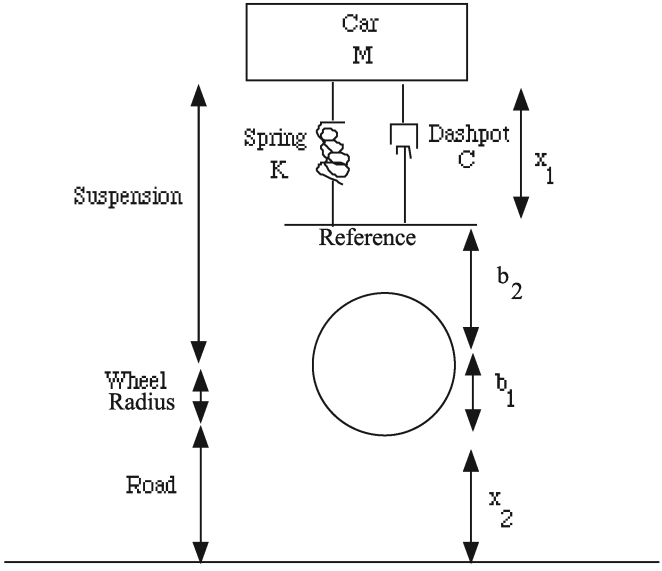

Fig. 5.20 Suspension system is represented by spring and dashpot.

Fig. 5.20 is not supposed to transmit any force to the mass as long as it is at rest, but, as soon as the mass moves, the damping force of the dashpot is proportional to the velocity and directed opposite to it. The quantity C is known as the damping constant or as the coefficient of viscous damping.

From Newton's second law, where force is mass times acceleration, we obtain from Fig. 5.20

$$M(\ddot{x}_1 + \ddot{b}_2 + \ddot{b}_1 + \ddot{x}_2) = -C\dot{x}_1 - Kx_1$$

Because the radius of the wheel b_1 is a constant, its first and second derivative must be zero. In addition, because b_2 is a constant, its second derivative must also be zero. By dividing both sides of the preceding equation by the mass of the car we obtain

$$\ddot{x}_1 + \frac{C}{M}\dot{x}_1 + \frac{K}{M}x_1 = -\ddot{x}_2$$

For convenience we can express the damping ζ and natural frequency ω_n of the suspension mechanism in terms of the spring constant and coefficient of viscous damping. Therefore, we can define

$$2\zeta\omega_n = \frac{C}{M}$$
$$\omega_n^2 = \frac{K}{M}$$

Using the preceding definitions, the second-order differential equation can now be rewritten in more convenient form as

$$\ddot{x}_1 = -2\zeta\omega_n\dot{x}_1 - \omega_n^2 x_1 - \ddot{x}_2$$

We have already mentioned that the bumpy road is represented by

$$x_2 = A_2 \sin \omega t$$

Therefore, the first and second derivatives of x_2 can be obtained by inspection of the preceding equation as

$$\dot{x}_2 = A_2\omega \cos \omega t$$
$$\ddot{x}_2 = -A_2\omega^2 \sin \omega t$$

The last equation is all we need for the second-order differential equation of the effective suspension length x_1.

Let us now consider a numerical example in which the amplitude and frequency of the sinusoidal road are 0.1 ft and 6.28 rad/s, respectively, while the initial suspension length is 1.5 ft. Initially we assume that $x_1 = 0.25$ ft and that $b_2 = 1.25$ ft. The wheel radius is considered to be 1 ft. We now have enough

information to integrate the second-order differential equation. Listing 5.4 presents the program that numerically integrates the suspension differential equation using the second-order Runge–Kutta integration technique described in Chapter 1. The structure of this program is identical to that of Listing 1.7 in Chapter 1. Printed out are the length of the suspension X1, the height of the road X2, and the height of the car from the reference DIST.

Listing 5.4 Simulation that integrates the suspension differential equation

```
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
WN=6.28*.1
W=6.28*1.
Z=.7
A2=.1
X1=.25
B2=1.25
X1D=0.
B1=1.
T=0.
S=0.
H=.001
WHILE(T<=20.)
        S=S+H
        X1OLD=X1
        X1DOLD=X1D
        X2=A2*SIN(W*T)
        X2D=A2*W*COS(W*T)
        X2DD=-A2*W*W*SIN(W*T)
        X1DD=-2.*Z*WN*X1D-WN*WN*X1-X2DD
        X1=X1+H*X1D
        X1D=X1D+H*X1DD
        T=T+H
        X2=A2*SIN(W*T)
        X2D=A2*W*COS(W*T)
        X2DD=-A2*W*W*SIN(W*T)
        X1DD=-2.*Z*WN*X1D-WN*WN*X1-X2DD
        X1=.5*(X1OLD+X1+H*X1D)
        X1D=.5*(X1DOLD+X1D+H*X1DD)
        IF(S>=.09999)THEN
                S=0.
                DIST=X1+X2+B1+B2
                SUSP=X1+B2
                WRITE(9,*)T,SUSP,X2,DIST
                WRITE(1,*)T,SUSP,X2,DIST
        ENDIF
END  DO
PAUSE
CLOSE(1)
END
```

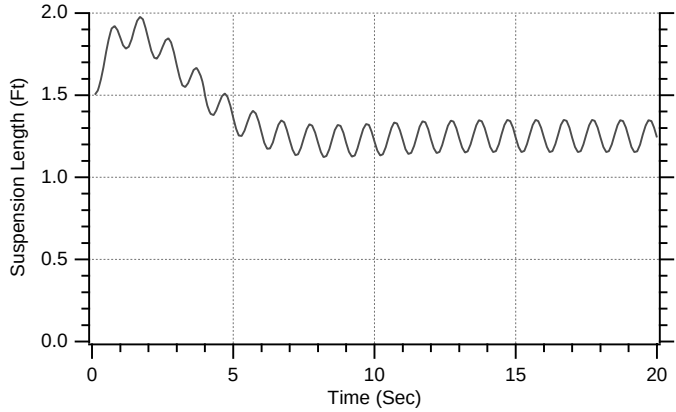

Fig. 5.21 Suspension oscillates at the road frequency.

The nominal case of Listing 5.4 was run. We can see from Fig. 5.21 that the suspension length oscillates at the frequency of the road. The amplitude of the suspension oscillations are approximately the amplitude of the road bumps. However, we can see from Fig. 5.22 that the suspension oscillations cancel out the road oscillations because the car height $x_1 + b_2 + b_1 + x_2$ above the reference appears to be approximately a constant after an initial transient period. In other words, it appears that everything is working properly because the suspension is protecting the car from the bumpy ride.

Kalman Filter for Suspension System

Recall from Fig. 5.19 that we have a sensor that is taking measurements of the distance from the car body to a reference. We desire to estimate x_1 based on noisy

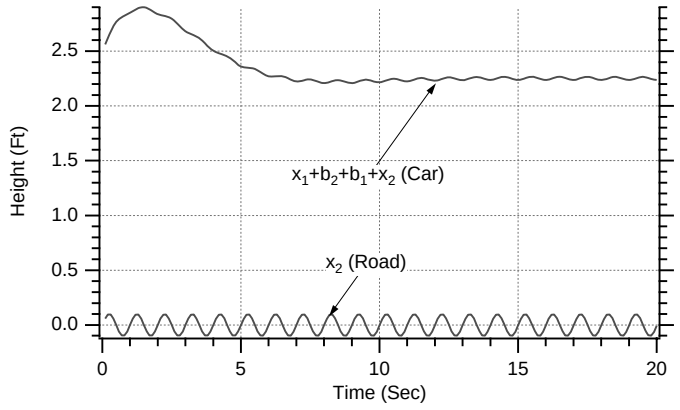

Fig. 5.22 Suspension enables the car to have a smooth ride.

measurements of the distance from the car body to the reference. Before we can build a Kalman filter for the estimation process we must first express the second-order differential equation for the suspension as two first-order equations in state-space form as

$$\begin{bmatrix} \dot{x}_1 \\ \ddot{x}_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \begin{bmatrix} x_1 \\ \dot{x}_1 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} \ddot{x}_2$$

The actual measurement is a function of the suspension length, road height, and wheel radius. The true measurement equation can be written as

$$\text{Meas} = x_1 + x_2 + b_1 + b_2 + v$$

where v is the measurement noise. To fit the Kalman-filtering equations where the measurement must be a linear function of the states, we can define a pseudo-measurement as

$$x_1^* = \text{Meas} - x_2 - b_1 - b_2 = x_1 + v = [1 \quad 0] \begin{bmatrix} x_1 \\ \dot{x}_1 \end{bmatrix} + v$$

where we are assuming prior knowledge of x_2, b_1, and b_2. From the preceding equation we can see that the measurement matrix is given by

$$\boldsymbol{H} = [1 \quad 0]$$

The systems dynamics matrix can be obtained directly from the state-space equation as

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

Therefore, the fundamental matrix in Laplace transform notation is given by

$$\Phi(s) = (sI - F)^{-1} = \begin{bmatrix} s & -1 \\ \omega_n^2 & s + 2\zeta\omega_n \end{bmatrix}^{-1}$$

Taking the inverse of a two-dimensional square matrix yields

$$\Phi(s) = \frac{1}{s^2 + 2\zeta\omega_n s + \omega_n^2} \begin{bmatrix} s + 2\zeta\omega_n & 1 \\ -\omega_n^2 & s \end{bmatrix}$$

If we define

$$a = -\zeta\omega_n$$

$$b = \omega_n\sqrt{1 - \zeta^2}$$

then the denominator in the expression for the fundamental matrix becomes

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = (s-a)^2 + b^2$$

From inverse Laplace transform tables[2] we know that the Laplace transforms of interest can be converted to the time domain according to

$$\frac{1}{(s-a)^2 + b^2} = \frac{e^{at}\sin bt}{b}$$
$$\frac{s}{(s-a)^2 + b^2} = \frac{e^{at}(a\sin bt + b\cos bt)}{b}$$

Therefore, we can find the fundamental matrix in the time domain to be

$$\Phi(t) = \begin{bmatrix} \dfrac{e^{at}(-a\sin bt + b\cos bt)}{b} & \dfrac{e^{at}\sin bt}{b} \\ \dfrac{-\omega_n^2 e^{at}\sin bt}{b} & \dfrac{e^{at}(a\sin bt + b\cos bt)}{b} \end{bmatrix}$$

The discrete fundamental matrix can be found from the preceding expression by simply replacing time with the sampling time or

$$\Phi_k = \begin{bmatrix} \dfrac{e^{aT_s}(-a\sin bT_s + b\cos bT_s)}{b} & \dfrac{e^{aT_s}\sin bT_s}{b} \\ \dfrac{-\omega_n^2 e^{aT_s}\sin bT_s}{b} & \dfrac{e^{aT_s}(a\sin bT_s + b\cos bT_s)}{b} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix}$$

From the state-space equation we can also see that

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

The discrete G matrix can be found from the continuous G matrix according to

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

Strictly speaking, the preceding equation is only valid if the deterministic input is constant between the sampling instants. Because the input is a high-frequency sinusoid, we know that this approximation is not good. However, we will proceed

nonetheless and correct any resultant errors by using process noise. Substitution of the appropriate matrices into the preceding integral equation yields

$$\boldsymbol{G}_k = \int_0^{T_s} \begin{bmatrix} \dfrac{e^{a\tau}(-a\sin b\tau + b\cos b\tau)}{b} & \dfrac{e^{a\tau}\sin b\tau}{b} \\ \dfrac{-\omega_n^2 e^{a\tau}\sin b\tau}{b} & \dfrac{e^{a\tau}(a\sin b\tau + b\cos b\tau)}{b} \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} \mathrm{d}\tau$$

Matrix multiplication simplifies the preceding expression to

$$\boldsymbol{G}_k = \int_0^{T_s} \begin{bmatrix} \dfrac{e^{a\tau}\sin b\tau}{b} \\ -\dfrac{e^{a\tau}(a\sin b\tau + b\cos b\tau)}{b} \end{bmatrix} \mathrm{d}\tau$$

From integration tables[2] we know that

$$\int e^{ax}\sin bx\,\mathrm{d}x = \frac{e^{ax}(a\sin bx - b\cos bx)}{a^2+b^2}$$
$$\int e^{ax}\cos bx\,\mathrm{d}x = \frac{e^{ax}(a\cos bx + b\sin bx)}{a^2+b^2}$$

Therefore, the discrete G matrix becomes

$$\boldsymbol{G}_k = \begin{bmatrix} -\dfrac{e^{aT_s}(a\sin bT_s - b\cos bT_s) + b}{b(a^2+b^2)} \\ -\dfrac{e^{aT_s}\sin bT_s}{b} \end{bmatrix} = \begin{bmatrix} G_1 \\ G_2 \end{bmatrix}$$

The discrete linear Kalman-filtering equation is given by

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

Substitution of the appropriate matrices into the preceding equation yields

$$\begin{bmatrix} \hat{x}_{1_k} \\ \hat{\dot{x}}_{1_k} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \begin{bmatrix} \hat{x}_{1_{k-1}} \\ \hat{\dot{x}}_{1_{k-1}} \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \ddot{x}_2 + \begin{bmatrix} K_{1_k} \\ K_{2_k} \end{bmatrix} \left[x^*_{1_k} - [1 \quad 0] \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \begin{bmatrix} \hat{x}_{1_{k-1}} \\ \hat{\dot{x}}_{1_{k-1}} \end{bmatrix} - [1 \quad 0] \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \ddot{x}_2 \right]$$

After multiplying out the terms of the preceding matrix difference equation, we obtain the following scalar equations for the Kalman filter

$$\text{Res}_k = x^*_{1_k} - \Phi_{11}\hat{x}_{1_{k-1}} - \Phi_{12}\hat{\dot{x}}_{1_{k-1}} - G_1\ddot{x}_2$$

$$\hat{x}_{1_k} = \Phi_{11}\hat{x}_{1_{k-1}} + \Phi_{12}\hat{\dot{x}}_{1_{k-1}} + G_1\ddot{x}_2 + K_{1_k}\text{Res}_k$$

$$\hat{\dot{x}}_{1_k} = \Phi_{21}\hat{x}_{1_{k-1}} + \Phi_{22}\hat{\dot{x}}_{1_{k-1}} + G_2\ddot{x}_2 + K_{2_k}\text{Res}_k$$

Listing 5.5 presents the code for the Kalman filter that is attempting to measure the suspension parameters based on measurements of the car's distance from the reference. Both the single-run errors in the estimates of the filter states plus the covariance matrix predictions are printed out every sampling interval.

Listing 5.5 Using a Kalman filter for measuring suspension length

```
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),G(2,1)
    INTEGER ORDER
    OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
    OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
    ORDER=2
    TF=20.
    SIGX=.1
    TS=.01
    WN=6.28*.1
    W=6.28*1.
    Z=.7
    A=-Z*WN
    B=WN*SQRT(1.-Z*Z)
    A2=.1
    X1=.25
    B2=1.25
    X1D=0.
    B1=1.
    T=0.
    S=0.
    H=.001
    X2DDOLD=0.
    DO 14 I=1,ORDER
    DO 14 J=1,ORDER
```

(*continued*)

Listing 5.5 (*Continued*)

```
    PHI(I,J)=0.
    P(I,J)=0.
    Q(I,J)=0.
    IDN(I,J)=0.
14  CONTINUE
    IDN(1,1)=1.
    IDN(2,2)=1.
    PHI(1,1)=EXP(A*TS)*(-A*SIN(B*TS)+B*COS(B*TS))/B
    PHI(1,2)=EXP(A*TS)*SIN(B*TS)/B
    PHI(2,1)=-WN*WN*EXP(A*TS)*SIN(B*TS)/B
    PHI(2,2)=EXP(A*TS)*(A*SIN(B*TS)+B*COS(B*TS))/B
    HMAT(1,1)=1.
    HMAT(1,2)=0.
    G(1,1)=-(EXP(A*TS)*(A*SIN(B*TS)-B*COS(B*TS))+B)/(B*(A*A+B*B))
    G(2,1)=-EXP(A*TS)*SIN(B*TS)/B
    CALL  MATTRN(PHI,ORDER,ORDER,PHIT)
    CALL  MATTRN(HMAT,1,ORDER,HT)
    P(1,1)=9999999.
    P(2,2)=9999999.
    Q(2,2)=0.
    X1H=X1
    X1DH=X1D
    RMAT(1,1)=SIGX**2
    WHILE(T < =20.)
            S=S+H
            X1OLD=X1
            X1DOLD=X1D
            X2=A2*SIN(W*T)
            X2D=A2*W*COS(W*T)
            X2DD=-A2*W*W*SIN(W*T)
            X1DD=-2.*Z*WN*X1D-WN*WN*X1-X2DD
            X1=X1+H*X1D
            X1D=X1D+H*X1DD
            T=T+H
            X2=A2*SIN(W*T)
            X2D=A2*W*COS(W*T)
            X2DD=-A2*W*W*SIN(W*T)
            X1DD=-2.*Z*WN*X1D-WN*WN*X1-X2DD
            X1=.5*(X1OLD+X1+H*X1D)
            X1D=.5*(X1DOLD+X1D+H*X1DD)
            IF(S>=(TS-.00001))THEN
                    S=0.
                    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)
```

(*continued*)

Listing 5.5 (*Continued*)

```
                        CALL  MATMUL(HM,1,ORDER,HT,ORDER,1,HMHT)
                        CALL  MATADD(HMHT,1,1,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,SIGX)
                        XMEAS=X1+B1+X2+B2+XNOISE
                        XS=XMEAS-X2-B1-B2
                        RES=XS-PHI(1,1)*X1H-PHI(1,2)*X1DH-G(1,1)*
                          X2DDOLD
                        X1HOLD=X1H
                        X1H=PHI(1,1)*X1H+PHI(1,2)*X1DH+G(1,1)*
                          X2DDOLD+K(1,1)*RES
                        X1DH=PHI(2,1)*X1HOLD+PHI(2,2)*X1DH+G(2,1)*
                          X2DDOLD+K(2,1)*RES
                        ERRX1=X1-X1H
                        SP11=SQRT(P(1,1))
                        ERRX1D=X1D-X1DH
                        SP22=SQRT(P(2,2))
                        X2DDOLD=X2DD
                        WRITE(9,*)T,X1,X1H,X1D,X1DH
                        WRITE(1,*)T,X1,X1H,X1D,X1DH
                        WRITE(2,*)T,ERRX1,SP11,-SP11,ERRX1D,SP22,-SP22
                ENDIF
        END  DO
        PAUSE
        CLOSE(1)
        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
```

The nominal case of Listing 5.5 was run in which there was no process noise. We can see from Figs. 5.23 and 5.24 that it appears that the filter is working well because the estimates of the suspension parameter x_1 and its derivative appear to be very close to the actual values.

To see if the Kalman filter is performing as expected we must examine the errors in the estimates of both filter states. We expect that the errors in the estimates should go to zero because there is no process noise. However, Figs. 5.25 and 5.26 show that the single-run simulation results of the errors in the

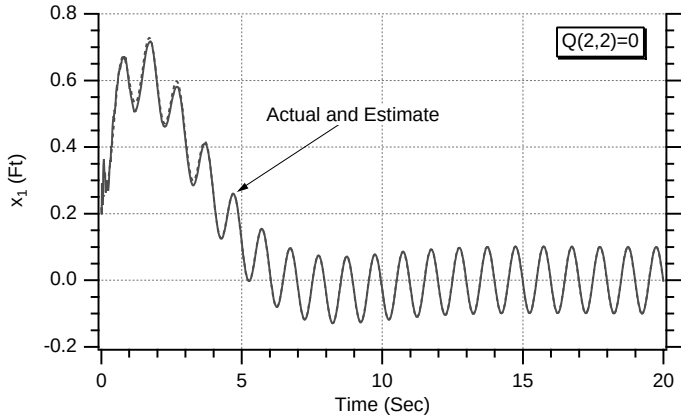

Fig. 5.23 Kalman filter provides excellent estimates of the first state of the suspension.

estimates oscillate, rather than go to zero, as more measurements are made. These results do not match the theoretical predictions of the covariance matrix, indicating that something is wrong.

To see if a mistake was made in deriving Φ_k and $\boldsymbol{G}_k$, another experiment was conducted. The filter propagation was tested with the filter states initialized perfectly and the Kalman gains set to zero. In other words, the filter simply coasts and the modified filtering equations of Listing 5.5 become

```
X1H=PHI(1,1)*X1H+PHI(1,2)*X1DH+G(1,1)*X2DDOLD
X1DH=PHI(2,1)*X1HOLD+PHI(2,1)*X1DH+G(2,1)*X2DDOLD
```

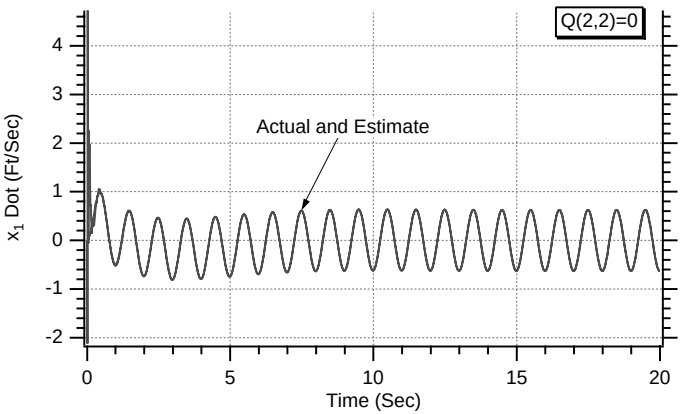

Fig. 5.24 Kalman filter provides excellent estimates of the derivative of the first state of the suspension.

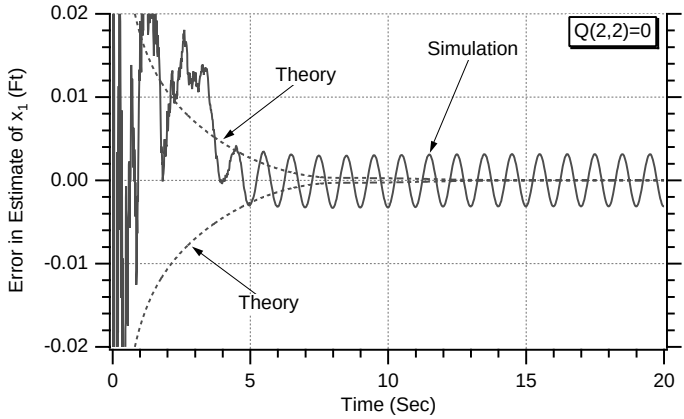

Fig. 5.25 Error in the estimate of first state does not agree with covariance matrix predictions.

We can see from Fig 5.27 that the error in the estimate of the first state still oscillates even though the filter was perfectly initialized. Under these circumstances the error in the estimate of the first state should have been close to zero. These results indicate that there is a mistake in either or both Φ_k and $\boldsymbol{G}_k$.

To isolate the cause of the problem, the simulation was further simplified by setting X2DD to zero. By doing this any errors in $\boldsymbol{G}_k$ will disappear, and we will only be checking the fundamental matrix Φ_k. We can see from Fig. 5.28 that when X2DD is set to zero the errors in the estimate of x_1 stop oscillating and become many orders of magnitude smaller than they were before indicating that the fundamental matrix is correct.

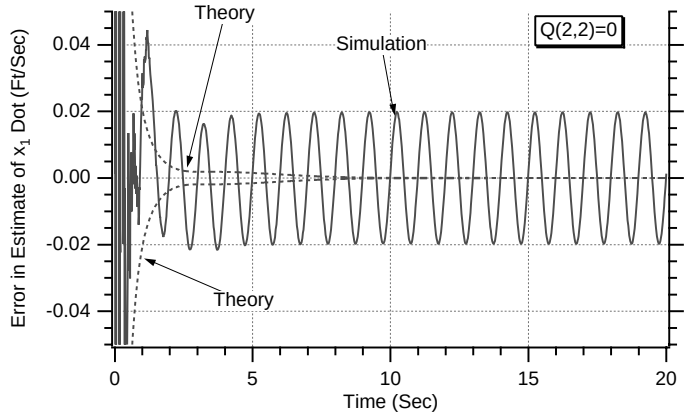

Fig. 5.26 Error in the estimate of second state does not agree with covariance matrix predictions.

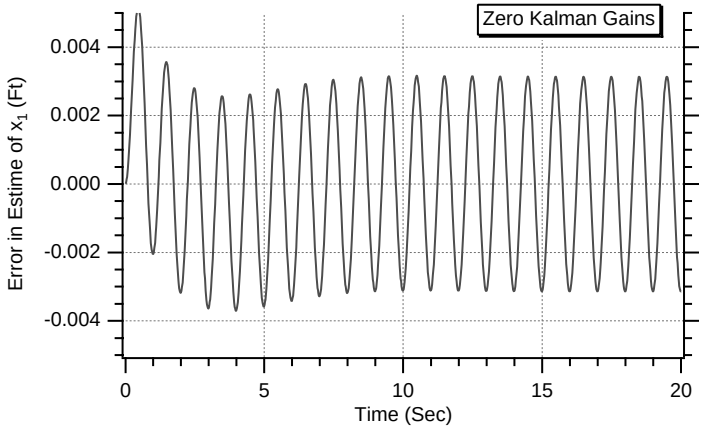

Fig. 5.27 Kalman filter does not coast properly when filter is perfectly initialized.

Strictly speaking $\boldsymbol{G}_k$ is only valid if the deterministic input X2DD is constant between sampling instants. To see if a mistake was made in deriving $\boldsymbol{G}_k$, another experiment was conducted in which the deterministic input was set to unity. We can see from Fig. 5.29 that now, because the deterministic input is indeed constant between sampling instants, the error in the estimate of the first state stops oscillating. These results prove that $\boldsymbol{G}_k$ is correct for the assumptions made. However $\boldsymbol{G}_k$ is not correct for this problem.

At this juncture we have two possible ways to go. We could derive a new discrete G matrix based on the fact that the deterministic input is sinusoidal between sampling instants, or we could simply add process noise to the filter to account for the fact that we do not have a perfect model of the real world. Because the first method requires a great deal of mathematics while the second method

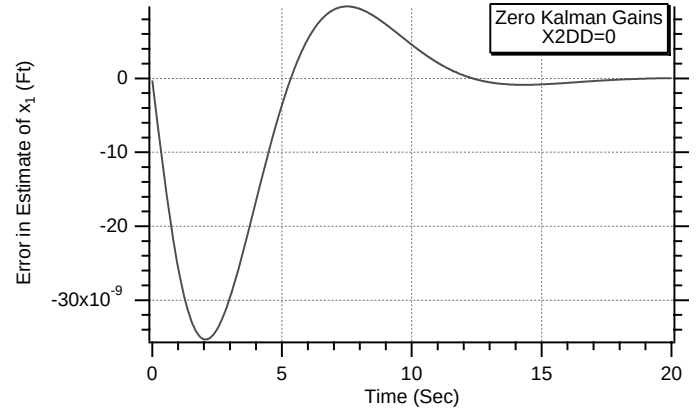

Fig. 5.28 Fundamental matrix appears to be correct.

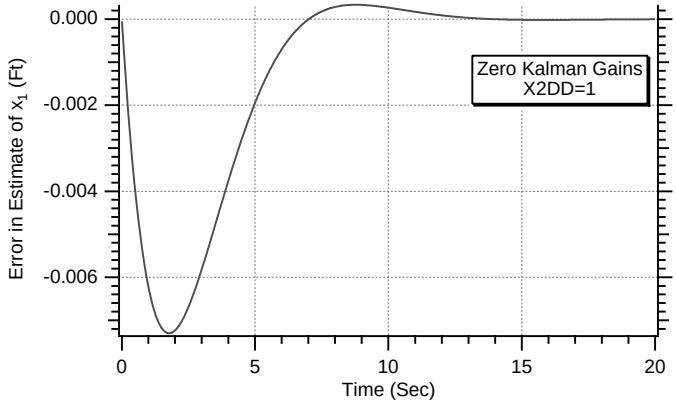

Fig. 5.29 Discrete G matrix is correct if deterministic input is constant.

only requires a one-line program change, the second method was chosen. When process noise is added, we can now see from Figs. 5.30 and 5.31 that the errors in the estimates of the first and second states are within the theoretical error bounds, indicating that the Kalman filter is now working properly.

The suspension length is $x_1 + b_2$, with b_2 being fixed at 1.25 ft. We know from Fig. 5.23 that x_1 varies between ± 0.1 ft and that the standard deviation of the measurement noise is 0.1 ft. Therefore, the suspension length varies between 1.24 and 1.26 ft. We can see from Fig. 5.30 that we have effectively reduced the effect of the measurement noise by nearly an order of magnitude because the error in the estimate of x_1 is approximately ± 0.015 ft. We know from Fig. 5.24 that the actual derivative of x_1 varies between ± 0.75 ft/s. From Fig. 5.31 we can see that

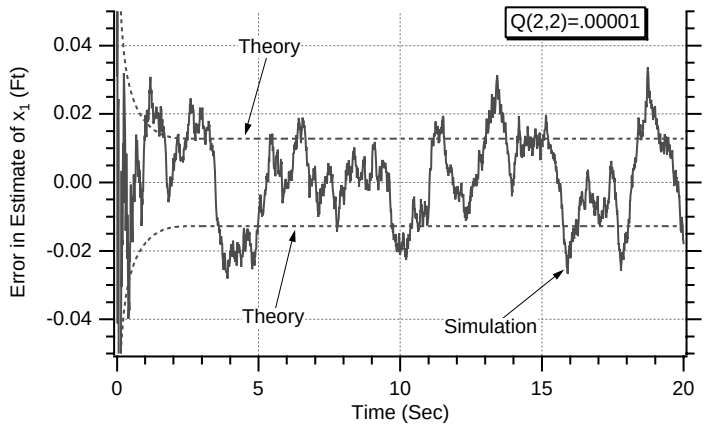

Fig. 5.30 Addition of process noise ensures that errors in the estimate of the first state are within the theoretical error bounds.

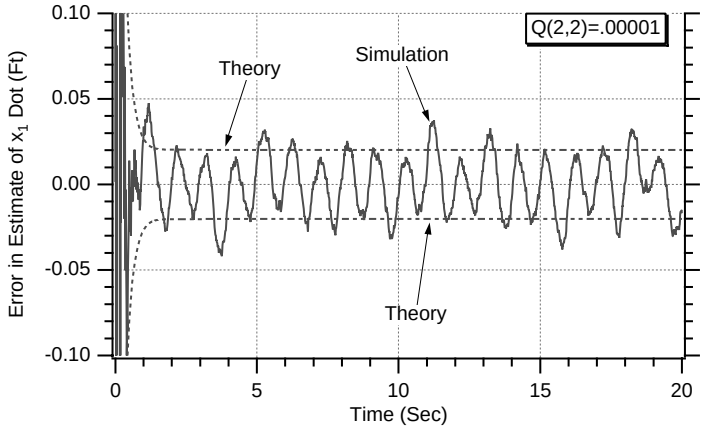

Fig. 5.31 Addition of process noise ensures that errors in the estimate of the second state are within the theoretical error bounds.

the Kalman filter is able to estimate the derivative of x_1 to within $\pm.025$ ft/s. Thus, we can see that even with process noise the Kalman filter is very effective.

Summary

In this chapter we have provided examples of various types of Kalman filters that could be used when the measurement signal was not a polynomial. If possible, we saw that it was important to have an accurate fundamental matrix. However, we also saw that the main importance of the fundamental matrix was in the propagation of the state estimates for the Kalman filter. Using an exact fundamental matrix was not as important for the computation of the Kalman gains. We also demonstrated that adding process noise to the filter was often the engineering fix for accounting for errors in the fundamental matrix or any other errors that could cause the filter to be mismatched from the real world.

References

[1]Schwarz, R. J., and Friedland, B., *Linear Systems*, McGraw–Hill, New York, 1965, pp. 106–128.

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

[3]Den Hartog, J. P., *Mechanical Vibrations*, Dover, New York, 1984, pp. 24–29.

[4]Maymon, G., *Some Engineering Applications in Random Vibrations and Random Structures*, Progress in Astronautics and Aeronautics, AIAA, Reston, VA, 1998, pp. 1–12.