Chapter 3

Recursive Least-Squares Filtering

Introduction

IN THE preceding chapter we saw how the method of least squares could be applied to estimating a signal based upon noisy measurements. This estimation process is also sometimes called filtering. We also observed that the least-squares technique was a batch-processing method because all of the measurements had to be taken before any estimates of the best polynomial coefficients could be made. In addition, a matrix inverse had to be evaluated as part of the required computation. The dimension of the matrix inverse was proportional to the order of the polynomial used to best fit the measurements in the least-squares sense. In this chapter we will see how the batch-processing method of least squares of Chapter 2 can be made recursive. The resulting recursive least-squares filter does not involve taking matrix inverses. Because the new least-squares filter is recursive, estimates are available as soon as measurements are taken. The simple nature of the calculations involved make recursive least-squares filtering ideal for digital computer implementation.

Making Zeroth-Order Least-Squares Filter Recursive

We have shown in the preceding chapter that if we were trying to fit a zeroth-order polynomial or constant to a set of measurement data the best estimate (i.e., minimize sum of squares of the difference between the measurement and estimate) can be expressed as

$$\hat{x}_k = a_0 = \frac{\sum_{i=1}^{k} x_i^*}{k}$$

where x_i^* is the ith measurement and k is the number of measurements taken. In other words, we simply add up the measurements and divide by the number of

measurements taken to find the best estimate. By changing subscripts we can rewrite the preceding expression as

$$\hat{x}_{k+1} = \frac{\sum_{i=1}^{k+1} x_i^*}{k+1}$$

Expanding the numerator yields

$$\hat{x}_{k+1} = \frac{\sum_{i=1}^{k} x_i^* + x_{k+1}^*}{k+1}$$

Because

$$\sum_{i=1}^{k} x_i^* = k\hat{x}_k$$

by substitution we can say that

$$\hat{x}_{k+1} = \frac{k\hat{x}_k + x_{k+1}^*}{k+1}$$

Without changing anything we can add and subtract the preceding state estimate to the numerator of the preceding equation, yielding

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

Therefore, we can rewrite the preceding expression as

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

Because we would like k to start from one rather than zero, we can also rewrite the preceding expression, by changing subscripts, to

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

The preceding expression is now in the recursive form we desire because the new estimate simply depends on the old estimate plus a gain (i.e., $1/k$ for the zeroth-order filter) times a residual (i.e., current measurement minus preceding estimate).

Properties of Zeroth-Order or One-State Filter

We have just seen in the preceding section that the form of the zeroth-order or one-state recursive least-squares filter is given by

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

where the gain of the filter is

$$K_{1_k} = \frac{1}{k} \qquad k = 1, 2, \ldots, n$$

and the residual (i.e., difference between the present measurement and the preceding estimate) is given by

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

We can see that the filter gain for the zeroth-order recursive least-squares filter is unity for the first measurement (i.e., $k = 1$) and eventually goes to zero as more measurements are taken.

To better understand how the preceding formulas apply, let us reconsider the four measurement examples of the preceding chapter. For convenience Table 3.1 repeats the sample measurement data of Chapter 2.

Because initially $k = 1$, the first gain of the recursive zeroth-order least-squares filter is computed from

$$K_{1_1} = \frac{1}{k} = \frac{1}{1} = 1$$

To apply the other zeroth-order recursive least-squares filter formulas, we need to have an initial condition for our first estimate in order to get started. For now let us assume that we have no idea how to initialize the filter, and so we simply set the initial estimate to zero or

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

We now calculate the residual as

$$\text{Res}_1 = x_1^* - \hat{x}_0 = 1.2 - 0 = 1.2$$

Table 3.1 Sample measurement data

k	$(k-1)T_s$	x_k^*
1	0	1.2
2	1	0.2
3	2	2.9
4	3	2.1

and the new estimate becomes

$$\hat{x}_1 = \hat{x}_0 + K_{1_1}\text{Res}_1 = 0 + 1*1.2 = 1.2$$

We have now completed the first cycle of the recursive equations. For the next cycle with $k = 2$, the next gain is computed as

$$K_{1_2} = \frac{1}{k} = \frac{1}{2} = 0.5$$

whereas the next residual becomes

$$\text{Res}_2 = x_2^* - \hat{x}_1 = 0.2 - 1.2 = -1$$

Therefore, the second estimate of the recursive least-squares filter is calculated as

$$\hat{x}_2 = \hat{x}_1 + K_{1_2}\text{Res}_2 = 1.2 + 0.5*(-1) = 0.7$$

The calculations for processing the third measurement (i.e., $k = 3$) can be computed as

$$K_{1_3} = \frac{1}{k} = \frac{1}{3} = 0.333$$
$$\text{Res}_3 = x_3^* - \hat{x}_2 = 2.9 - 0.7 = 2.2$$
$$\hat{x}_3 = \hat{x}_2 + K_{1_3}\text{Res}_3 = 0.7 + 0.333*2.2 = 1.43$$

while the calculations for the fourth measurement (i.e., $k = 4$) turn out to be

$$K_{1_4} = \frac{1}{k} = \frac{1}{4} = 0.25$$
$$\text{Res}_4 = x_4^* - \hat{x}_3 = 2.1 - 1.43 = 0.67$$
$$\hat{x}_4 = \hat{x}_3 + K_{1_4}\text{Res}_4 = 1.43 + 0.25*0.67 = 1.6$$

Note that the last estimate of the zeroth-order least-squares recursive filter is identical to the last estimate obtained from the zeroth-order least-squares fit of Chapter 2.

To demonstrate that the recursive least-squares filter estimates are independent of initial conditions, suppose the initial estimate of the zeroth-order least-squares recursive filter was 100 rather than zero or

$$\hat{x}_0 = 100$$

Then the remaining calculations for the first estimate are

$$\text{Res}_1 = \hat{x}_1^* - \hat{x}_0 = 1.2 - 100 = -98.8$$
$$\hat{x}_1 = \hat{x}_0 + K_{1_1}\text{Res}_1 = 100 + 1*(-98.8) = 1.2$$

which is exactly the same answer we obtained when the initial estimate was zero. In other words, *the zeroth-order recursive least-squares filter will yield the same answers regardless of initial conditions*.

Figure 3.1 compares the estimates from the zeroth-order batch-processing method of least squares from Chapter 2 with the zeroth-order recursive method of least squares of this chapter. As was mentioned before, the batch-processing least-squares estimates are only available after all of the measurements are taken (i.e., we solve for polynomial coefficients after all of the measurements are taken and then evaluate the polynomial at different points in time), whereas the recursive least-squares estimates are available as the measurements are being taken. We can see from Fig. 3.1 that after all of the measurements are taken both the batch-processing and recursive least-squares methods yield the same answers.

If the actual measurement data are simply a constant plus noise (i.e., zeroth-order signal plus noise) where the measurement noise is a zero-mean Gaussian process with variance σ_n^2, then a formula can also be derived that describes the variance of the error in the filter's estimate (i.e., variance of actual signal minus filter estimate). Recall that the recursive form of the zeroth-order least-squares filter is given by

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

Therefore, the error in the estimate must be the actual signal minus the estimate or

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

Recognizing that the measurement is simply the signal plus noise or

$$x_k^* = x_k + v_k$$

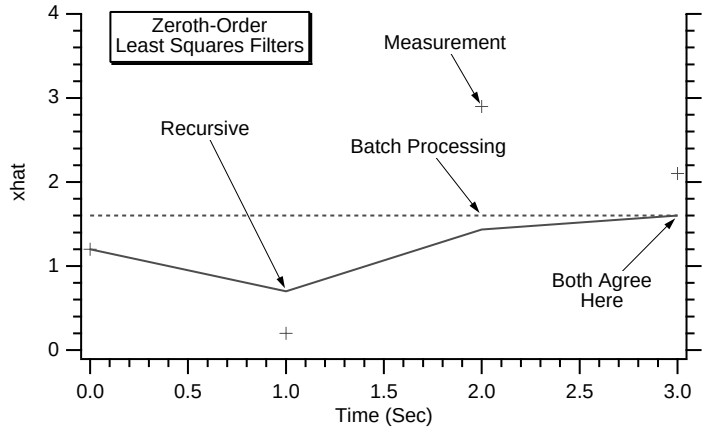

Fig. 3.1 Batch-processing and recursive least-squares methods yield the same answers after all measurements are taken.

we obtain for the error in the estimate as

$$x_k - \hat{x}_k = x_k - \hat{x}_{k-1} - \frac{1}{k}(x_k + v_k - \hat{x}_{k-1})$$

However, because the signal is a constant for the zeroth-order system, we can also say that

$$x_k = x_{k-1}$$

Therefore, substitution yields

$$x_k - \hat{x}_k = (x_{k-1} - \hat{x}_{k-1})\left(1 - \frac{1}{k}\right) - \frac{1}{k}v_k$$

After squaring both sides of the preceding equation, we obtain

$$(x_k - \hat{x}_k)^2 = (x_{k-1} - \hat{x}_{k-1})^2\left(1 - \frac{1}{k}\right)^2 - 2\left(1 - \frac{1}{k}\right)(x_{k-1} - \hat{x}_{k-1})\frac{v_k}{k} + \left(\frac{1}{k}v_k\right)^2$$

Taking expectations of both sides of the equation yields

$$E[(x_k - \hat{x}_k)^2] = E[(x_{k-1} - \hat{x}_{k-1})^2]\left(1 - \frac{1}{k}\right)^2 - 2\left(1 - \frac{1}{k}\right)E[(x_{k-1} - \hat{x}_{k-1})v_k]\frac{1}{k} + E\left[\left(\frac{1}{k}v_k\right)^2\right]$$

If we define the variance of the error in the estimate to be P_k and the variance of the measurement noise to be σ_n^2 and also assume that the noise is uncorrelated with the state or the estimate of the state, we can say that

$$E[(x_k - \hat{x}_k)^2] = P_k$$
$$E(v_k^2) = \sigma_n^2$$
$$E[(x_{k-1} - \hat{x}_{k-1})v_k] = 0$$

Substitution of the preceding three equations into the expectation equation yields the difference equation for the variance of the error in the estimate or

$$P_k = P_{k-1}\left(1 - \frac{1}{k}\right)^2 + \frac{\sigma_n^2}{k^2}$$

The preceding difference equation can be solved using Z-transform techniques, but it is far easier in this example to use engineering induction by substituting

different values of k. Substituting $k = 1, 2, 3$, and 4 into the preceding equation yields

$$P_1 = P_0\left(1 - \frac{1}{1}\right)^2 + \frac{\sigma_n^2}{1^2} = \sigma_n^2$$

$$P_2 = P_1\left(1 - \frac{1}{2}\right)^2 + \frac{\sigma_n^2}{2^2} = \sigma_n^2\frac{1}{4} + \frac{\sigma_n^2}{4} = \frac{\sigma_n^2}{2}$$

$$P_3 = P_2\left(1 - \frac{1}{3}\right)^2 + \frac{\sigma_n^2}{3^2} = \frac{\sigma_n^2}{2}\frac{4}{9} + \frac{\sigma_n^2}{9} = \frac{\sigma_n^2}{3}$$

$$P_4 = P_3\left(1 - \frac{1}{4}\right)^2 + \frac{\sigma_n^2}{4^2} = \frac{\sigma_n^2}{3}\frac{9}{16} + \frac{\sigma_n^2}{16} = \frac{\sigma_n^2}{4}$$

The trend in the preceding four equations is now obvious, and we can summarize these results by saying in general for the zeroth-order recursive least-squares filter that the variance of the error in the estimate is simply

$$P_k = \frac{\sigma_n^2}{k}$$

where σ_n^2 is the variance of the measurement noise and k is the number of measurements taken (i.e., $k = 1, 2, \ldots, n$).

If, on the other hand, the real signal is a first-order polynomial (one degree higher than the zeroth-order filter) or

$$x_k = a_0 + a_1 t = a_0 + a_1(k-1)T_s$$

the filter will not be able to track the signal, as we saw in Chapter 2. The resulting error in the estimate is known as truncation error ε_k and by definition is the difference between the true signal and the estimate or

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

Recall that we showed in Chapter 2 that the estimate of the zeroth-order least-squares filter was simply

$$\hat{x}_k = \frac{\sum_{i=1}^{k} x_i^*}{k}$$

In the noise-free case the measurement is the signal, and so the preceding equation can be rewritten as

$$\hat{x}_k = \frac{\sum_{i=1}^{k} x_i}{k} = \frac{\sum_{i=1}^{k} [a_0 + a_1(i-1)T_s]}{k} = \frac{a_0 \sum_{i=1}^{k} + a_1 T_s \sum_{i=1}^{k} i - a_1 T_s \sum_{i=1}^{k}}{k}$$

Because mathematical handbooks[1] tell us that

$$\sum_{i=1}^{k} = k$$

$$\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$$

we can eliminate summation signs in the expression for the estimate and obtain

$$\hat{x}_k = \frac{a_0 k + a_1 T_s[k(k+1)/2] - a_1 T_s k}{k} = a_0 + \frac{a_1 T_s}{2}(k-1)$$

Substituting the estimate into the equation for the error in the estimate yields

$$\varepsilon_k = x_k - \hat{x}_k = a_0 + a_1 T_s(k-1) - a_0 - \frac{a_1 T_s}{2}(k-1) = \frac{a_1 T_s}{2}(k-1)$$

which is the equation for the truncation error of the zeroth-order recursive least-squares filter.

Listing 3.1 shows how the zeroth-order recursive least-squares filter, along with the theoretical formulas, were programmed. In this example the listing indicates that the measurements of a constant signal ACT plus noise XS are taken every 0.1 s for 10 s. We can see from Listing 3.1 that the actual error in the estimate XHERR is computed and compared to the theoretical error in the estimate or plus and minus the square root of P_k (i.e., SP11 in Listing 3.1). As was the case in Chapter 2 for the zeroth-order measurement, nominally the simulation is set up so that the true signal is also a zeroth-order polynomial or constant with value unity (A0 = 1, A1 = 0), which is corrupted by noise with unity standard deviation (SIGNOISE = 1). We can also see that the preceding theoretical truncation error formula also appears in Listing 3.1.

A run was made with the nominal case of Listing 3.1, and the true signal, measurement, and filter estimate are compared in Fig. 3.2. We can see that the actual measurement is quite noisy because in this example the amplitude of the signal is equal to the standard deviation of the noise. However, we can also see that the filter estimate is quite good (i.e., solid curve in Fig. 3.2) and appears to be getting closer to the true signal as more measurements are taken.

For more exact filtering work the error in the estimate (i.e., difference between the true signal and estimate) is often used as a measure of performance. The error in the estimate can only be computed in a simulation where the actual signal is known. In the real world, where the filter must eventually work, the real signal or

Listing 3.1 Simulation for testing zeroth-order recursive least-squares filter

```
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)
    OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
    TS=0.1
    SIGNOISE=1.
    A0=1.
    A1=0.
    XH=0.
    XN=0.
    DO 10 T=0.,10.,TS
    XN=XN+1.
    CALL GAUSS(XNOISE,SIGNOISE)
    ACT=A0+A1*T
    XS=ACT+XNOISE
    XK=1./XN
    RES=XS-XH
    XH=XH+XK*RES
    SP11=SIGNOISE/SQRT(XN)
    XHERR=ACT-XH
    EPS=0.5*A1*TS*(XN-1)
    WRITE(9,*)T,ACT,XS,XH,XHERR,SP11,-SP11,EPS
    WRITE(1,*)T,ACT,XS,XH,XHERR,SP11,-SP11,EPS
 10 CONTINUE
    CLOSE(1)
    PAUSE
    END
C SUBROUTINE GAUSS IS SHOWN IN LISTING 1.8
```

truth is never available. A more complete discussion of this topic can be found in Chapter 14. Figure 3.3 compares the error in the estimate of x to the theoretical predictions of this section (i.e., square root of P_k or SP11 in Listing 3.1). We can see that the single-run results are within the theoretical bounds most of the time, indicating that theory and simulation appear to agree. Also, from the formula for the variance of the estimate (i.e., σ_n^2/k), we see that the error in the estimate will tend toward zero as the number of measurements k gets large. For an infinite number of measurements, the variance of the error in the estimate will be zero.

A noise-free case (SIGNOISE $= 0$) was also run to examine the effect of truncation error. In this example the actual signal is first-order (A0 $= 1$, A1 $= 2$), whereas the filter is zeroth-order. We can see from Fig. 3.4 that it is clear that the estimate from the zeroth-order recursive least-squares filter estimate is diverging from the true signal. Therefore, we can say that the zeroth-order recursive least-squares filter cannot track the first-order signal.

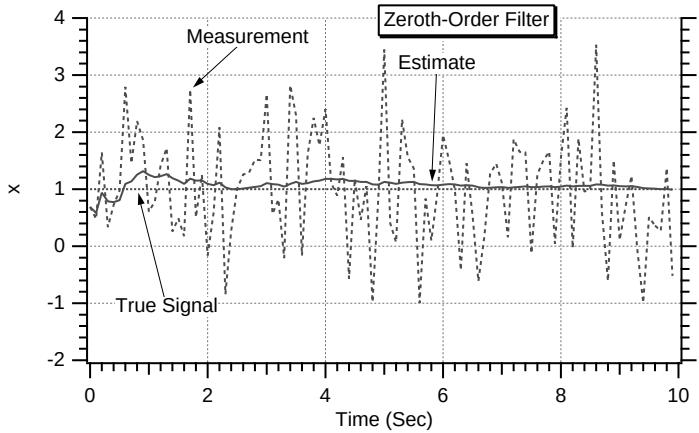

Fig. 3.2 Zeroth-order recursive least-squares filter is able to track zero-order polynomial plus noise.

Figure 3.5 displays the error in the estimate from simulation results and the theoretical estimate of the truncation error ε_k. For this particular numerical example the error caused by truncation is given by

$$\varepsilon_k = \frac{a_1 T_s}{2}(k-1) = 0.5*2*0.1(k-1) = 0.1(k-1)$$

where k goes from 1 to 100. We can see from Fig. 3.5 that the error in the estimate of x (i.e., XHERR in Listing 3.1) as a result of running Listing 3.1 and the truncation error formula (i.e., EPS in Listing 3.1) are identical, thus empirically validating the formula.

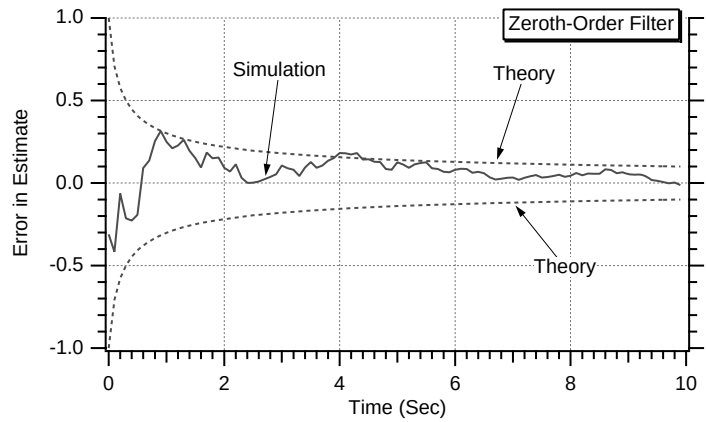

Fig. 3.3 Single-run simulation results agree with theoretical formula.

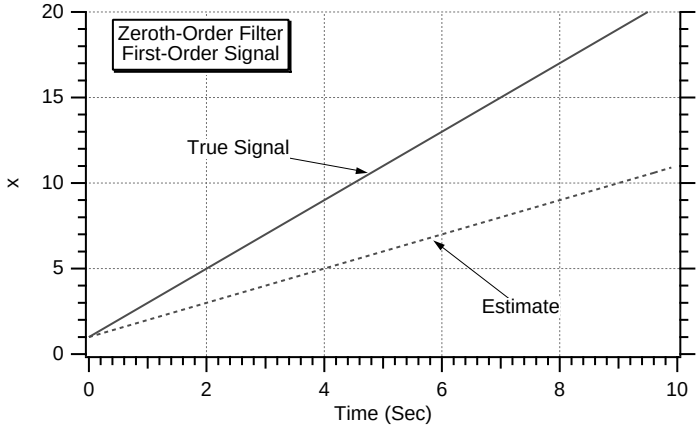

Fig. 3.4 Zeroth-order recursive least-squares filter is unable to track first-order polynomial.

Because the formula for the standard deviation of the error in the estimate caused by noise is given by

$$\sqrt{P_{11_k}} = \frac{\sigma_n}{\sqrt{k}}$$

and the error in the estimate caused by truncation error is

$$\varepsilon_k = 0.5a_1 T_s(k-1)$$

we can see that both errors behave in different ways. As already noted, as more measurements are taken k gets larger, and the error in the estimate caused by the

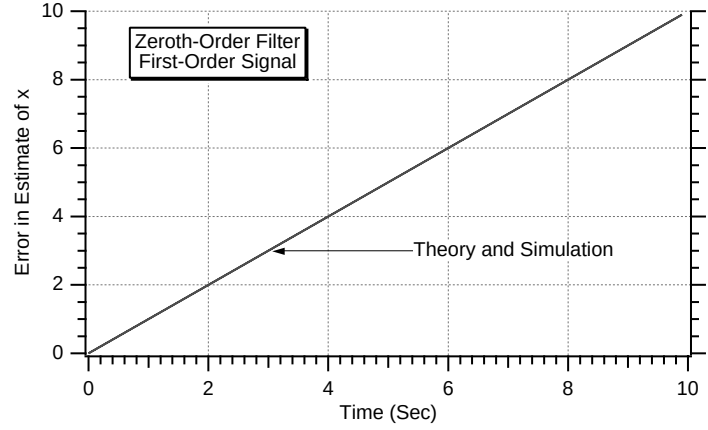

Fig. 3.5 Simulation results and truncation error formula are in excellent agreement.

measurement noise decreases while the error in the estimate caused by truncation error increases. In principle, for any particular numerical example there is an optimal value of k that will minimize the errors in the estimates caused by both measurement noise and truncation error.

If more runs were made in the case in which there was no truncation error, we would expect to find the computed error in the estimate of x to lie between the theoretical abounds approximately 68% of the time. To ensure that the simulated and theoretical errors in the estimates agree in this way, Listing 3.1 was slightly modified so that we could run repeated simulation trials, also known as the Monte Carlo method. The Monte Carlo version of Listing 3.1 appears in Listing 3.2. We

Listing 3.2 Monte Carlo simulation for testing zeroth-order recursive least-squares filter

```
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)
    OPEN(1,STATUS='UNKNOWN',FILE='DATFIL1')
    OPEN(2,STATUS='UNKNOWN',FILE='DATFIL2')
    OPEN(3,STATUS='UNKNOWN',FILE='DATFIL3')
    OPEN(4,STATUS='UNKNOWN',FILE='DATFIL4')
    OPEN(5,STATUS='UNKNOWN',FILE='DATFIL5')
    DO 11 K=1,5
    TS=0.1
    SIGNOISE=1.
    A0=1.
    A1=0.
    XH=0.
    XN=0.
    DO 10 T=0.,10.,TS
    XN=XN+1.
    CALL GAUSS(XNOISE,SIGNOISE)
    ACT=A0+A1*T
    XS=ACT+XNOISE
    XK=1./XN
    RES=XS-XH
    XH=XH+XK*RES
    SP11=SIGNOISE/SQRT(XN)
    XHERR=ACT-XH
    EPS=0.5*A1*TS*(XN-1)
    WRITE(9,*)T,XHERR,SP11,-SP11
    WRITE(K,*)T,XHERR,SP11,-SP11
 10 CONTINUE
    CLOSE(K)
 11 CONTINUE
    PAUSE
    END
C SUBROUTINE GAUSS IS SHOWN IN LISTING 1.8
```

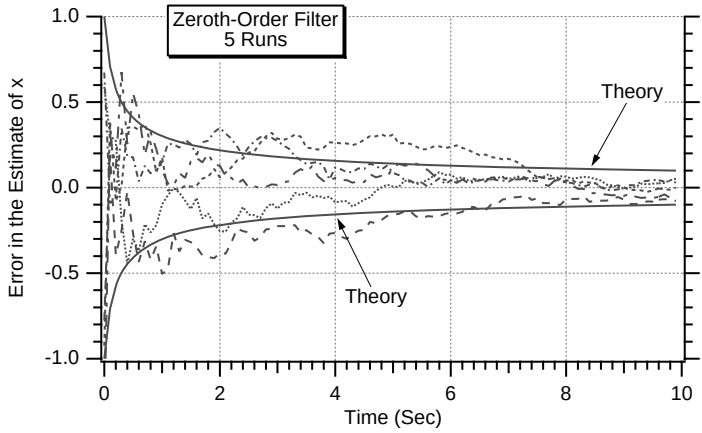

Fig. 3.6 Monte Carlo results lie within the theoretical bounds approximately 68% of the time.

can see that a 5 run "do loop" has been created, and the results of each run are written to files DATFIL1 through DATFIL5. The statements that were changed from Listing 3.1, in order to get a Monte Carlo version of the program, are highlighted in boldface in Listing 3.2.

A five-run Monte Carlo set was run with Listing 3.2. We can see from Fig. 3.6 that the errors in the estimate of x from each of the simulation trials appears, on average, to lie within the theoretical bounds approximately 68% of the time. These results indicate that the recursive zeroth-order least-squares filter appears to be behaving according to the theoretical predictions (i.e., formula for standard deviation of error in the estimate).

Properties of First-Order or Two-State Filter

Using techniques similar to the ones used in the first section of this chapter, the two gains of a first-order or two-state recursive least-squares filter can be shown to be[2]

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

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

We can see that, as was the case before, both filter gains eventually go to zero as more measurements are taken (i.e., k gets larger). If we think of the measurement as a position, then the estimates from this two-state filter will be position and velocity (i.e., derivative of position). In other words, position is the first state, whereas the derivative of position or velocity is the second state. For this filter a

residual is first formed, which is the difference between the present measurement and a projection of the preceding estimate to the current time or

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

The new filter estimates are a combination of the preceding state estimates projected forward to the current time plus a gain multiplied by the residual or

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

To understand better how these formulas apply, let us again reconsider the four measurement examples of the preceding chapter and the preceding section, where the measurements are

$$x_1^* = 1.2$$
$$x_2^* = 0.2$$
$$x_3^* = 2.9$$
$$x_4^* = 2.1$$

The calculations for the gains, residual, and states for each value of k are summarized, in detail, in Table 3.2. For each value of k, we compute gains, a residual, and the two-state estimates.

As was the case in the preceding section, *the last estimate of the first-order recursive least-squares filter (i.e., $\hat{x}_4 = 2.41$) is identical to the estimate obtained from the first-order least-squares fit of Chapter 2*.

Figure 3.7 compares the estimates from both the batch-processing method of least squares for the first-order system of Chapter 2 and the first-order recursive least-squares filter of this section. Recall that the first-order recursive least-squares filter state estimates are based on the calculations of Table 3.2. Again we can see from Fig. 3.7 that, after all of the measurements are taken, both the batch-processing and recursive least-squares filters yield the same answers. The recursive first-order filter estimate passes through the first two measurements.

If the measurement data are also a first-order polynomial and the measurement noise statistics are known, then formulas can also be derived, using techniques similar to those used on the zeroth-order filter of the preceding section, describing the variance of the error in the estimates in both states (i.e., P_{11} and P_{22}) of the first-order filter. The variance of the error in the estimate for the first and second states are given by[2]

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

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

Table 3.2 First-order recursive least-squares-filter calculations

Value of k	Calculations
$k = 1$	$K_{1_1} = \dfrac{2(2k-1)}{k(k+1)} = \dfrac{2(2*1-1)}{1*1+1)} = 1$
	$K_{2_k} = \dfrac{6}{k(k+1)T_s} = \dfrac{6}{1*(1+1)*1} = 3$
	$\text{Res}_1 = x_1^* - \hat{x}_0 - \hat{\dot{x}}_0 T_s = 1.2 - 0 - 0*1 = 1.2$
	$\hat{x}_1 = \hat{x}_0 + \hat{\dot{x}}_0 T_s + K_{1_1}\text{Res}_1 = 0 + 0*1 + 1*1.2 = 1.2$
	$\hat{\dot{x}}_1 = \hat{\dot{x}}_0 + K_{2_1}\text{Res}_1 = 0 + 3*1.2 = 3.6$
$k = 2$	$K_{1_2} = \dfrac{2(2k-1)}{k(k+1)} = \dfrac{2(2*2-1)}{2*2+1)} = 1$
	$K_{2_k} = \dfrac{6}{k(k+1)T_s} = \dfrac{6}{2*(2+1)*1} = 1$
	$\text{Res}_2 = x_2^* - \hat{x}_1 - \hat{\dot{x}}_1 T_s = 0.2 - 1.2 - 3.6*1 = -4.6$
	$\hat{x}_2 = \hat{x}_1 + \hat{x}_1 T_s + K_{1_2}\text{Res}_2 = 1.2 + 3.6*1 + 1*(-4.6) = 0.2$
	$\hat{\dot{x}}_2 = \hat{\dot{x}}_1 + K_{2_2}\text{Res}_2 = 3.6 + 1*(-4.6) = -1$
$k = 3$	$K_{1_3} = \dfrac{2(2k-1)}{k(k+1)} = \dfrac{2(2*3-1)}{3*(3+1)} = \dfrac{5}{6}$
	$K_{2_3} = \dfrac{6}{k(k+1)T_s} = \dfrac{6}{3*(3+1)*1} = 0.5$
	$\text{Res}_3 = x_3^* - \hat{x}_2 - \hat{\dot{x}}_2 T_s = 2.9 - 0.2 - (-1)*1 = 3.7$
	$\hat{x}_3 = \hat{x}_2 + \hat{\dot{x}}_2 T_s + K_{1_3}\text{Res}_3 = 0.2 + (-1)*1 + \dfrac{5*(3.7)}{6} = 2.28$
	$\hat{\dot{x}}_3 = \hat{\dot{x}}_2 + K_{2_3}\text{Res}_3 = -1 + 0.5*3.7 = 0.85$
$k = 4$	$K_{1_4} = \dfrac{2(2k-1)}{k(k+1)} = \dfrac{2(2*4-1)}{4*(4+1)} = 0.7$
	$K_{2_4} = \dfrac{6}{k(k+1)T_s} = \dfrac{6}{4*(4+1)*1} = 0.3$
	$\text{Res}_4 = x_4^* - \hat{x}_3 - \hat{\dot{x}}_3 T_s = 2.1 - 2.28 - 0.85*1 = -1.03$
	$\hat{x}_4 = \hat{x}_3 + \hat{\dot{x}}_3 T_s + K_{1_4}\text{Res}_4 = 2.28 + 0.85*1 + 0.7*(-1.03) = 2.41$
	$\hat{\dot{x}}_4 = \hat{\dot{x}}_3 + K_{2_4}\text{Res}_4 = 0.85 + 0.3*(-1.03) = 0.54$

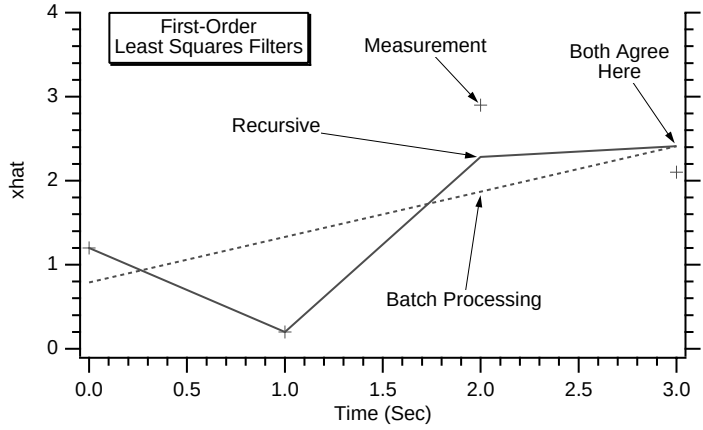

Fig. 3.7 First-order least-squares recursive and batch-processing least-squares filters yield the same answers after all measurements are taken.

where σ_n^2 is the variance of the measurement noise, k is the number of measurements taken (i.e., $k = 1, 2, \ldots, n$), and T_s is the sample time or time between measurements. As before, the variance of the errors in the estimates decrease as the number of measurements increases.

If, on the other hand, the real signal is a second-order polynomial (one degree higher than the first-order filter) or

$$x_k^* = a_0 + a_1 t + a_2 t^2 = a_0 + a_1(k-1)T_s + a_2(k-1)^2 T_s^2$$

the first-order filter will not be able to track the second-order signal as we saw in the preceding section. As was mentioned in the preceding section, the resulting error buildup caused by this lack of tracking ability is known as truncation error ε_k. The truncation errors for the states of the first-order recursive least-squares filter can be derived, using techniques similar to those of the preceding section on the zeroth-order filter, and are given by[2]

$$\varepsilon_k = \frac{1}{6} a_2 T_s^2 (k-1)(k-2)$$
$$\dot{\varepsilon}_k = a_2 T_s (k-1)$$

Listing 3.3 shows how the first-order recursive least-squares filter and second-order measurement signal, along with the theoretical formulas, were programmed. We can see that measurements of a first-order signal X plus noise XS is taken every 0.1 s for 10 s. We can see that the actual error in the estimate XHERR of the first state is compared to the theoretical error in the estimate of the first state (i.e., plus and minus the square root of P_{11}), whereas the actual error in the estimate of the second state XDHERR is compared to the theoretical error in the estimate of the second state (i.e., plus and minus the square root of P_{22}). Nominally the simulation is set so that the measurement is also a

Listing 3.3 Simulation for testing first-order recursive least-squares filter

```
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)
      TS=0.1
      SIGNOISE=5.
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      A0=3.
      A1=1.
      A2=0.
      XH=0.
      XDH=0.
      XN=0
      DO 10 T=0.,10.,TS
      XN=XN+1.
      CALL GAUSS(XNOISE,SIGNOISE)
      X=A0+A1*T+A2*T*T
      XD=A1+2*A2*T
      XS=X+XNOISE
      XK1=2*(2*XN-1)/(XN*(XN+1))
      XK2=6/(XN*(XN+1)*TS)
      RES=XS-XH-TS*XDH
      XH=XH+XDH*TS+XK1*RES
      XDH=XDH+XK2*RES
      IF(XN.EQ.1)THEN
        LET SP11=0
        LET SP22=0
      ELSE
        SP11=SIGNOISE*SQRT(2.*(2*XN-1)/(XN*(XN+1)))
        SP22=SIGNOISE*SQRT(12/(XN*(XN*XN-1)*TS*TS))
      ENDIF
      XHERR=X-XH
      XDHERR=XD-XDH
      EPS=A2*TS*TS*(XN-1)*(XN-2)/6
      EPSD=A2*TS*(XN-1)
      WRITE(9,*)T,X,XS,XH,XD,XDH
      WRITE(1,*)T,X,XS,XH,XD,XDH
      WRITE(2,*)T,XHERR,SP11,-SP11,EPS,XDHERR,SP22,-SP22,EPSD
 10   CONTINUE
      CLOSE(1)
      CLOSE(2)
      PAUSE
      END
C SUBROUTINE GAUSS IS SHOWN IN LISTING 1.8
```

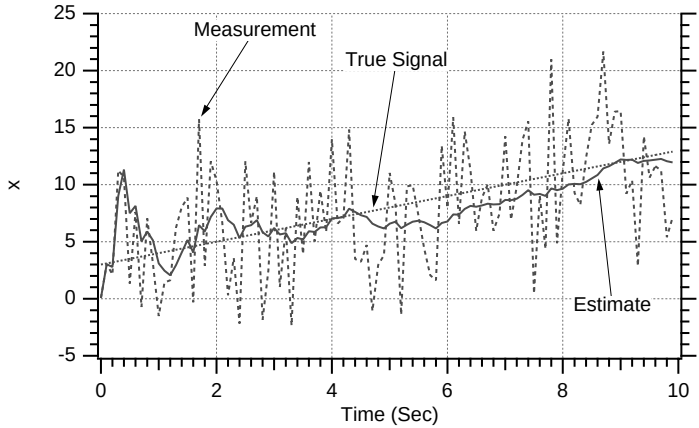

Fig. 3.8 First-order recursive least-squares filter is able to track first-order signal plus noise.

first-order polynomial with values A0 = 3 and A1 = 1 corrupted by noise with a standard deviation of five (SIGNOISE = 5). These values are identical to the values chosen for the sample first-order example in Chapter 2.

The nominal case of Listing 3.3 was run, and the true signal, measurement, and filter estimate of the first state are compared in Fig. 3.8. We can see that the actual measurement is quite noisy because the amplitude of the signal is comparable to the standard deviation of the noise. We can also see that the filter estimate of the true signal is quite good and is getting better as more measurements are taken. There is no problem in the state estimate tracking the actual signal because the order of the recursive filter is matched to the order of the true signal.

The true second state (i.e., derivative of first state x) for this example is simply

$$\dot{x}_k = 1$$

We can see from Fig. 3.9 that after a brief transient period the recursive first-order least-squares filter has no problem in estimating this state either.

Again, for more exact work the error in the estimate (i.e., difference between true signal and estimate) is used as a measure of performance. Figures 3.10 and 3.11 compare the error in the estimate of the first and second states of the first-order recursive least-squares filter to the theoretical predictions of this section (i.e., square root of P_{11} and P_{22}). We can see that the single-run simulation results are within the theoretical bounds most of the time, indicating that theory and simulation appear to agree. Even though the noise on the first state was of value five, the error in the estimate of x goes down by a factor of 5 after 100 measurements are taken.

We can see from Fig. 3.11 that the single-run simulation results for the error in the estimate of the second state (i.e., derivative of true signal minus second state) also lie within the theoretical error bounds, indicating that the filter is working properly. Figures 3.10 and 3.11 both indicate that the errors in the estimates of the

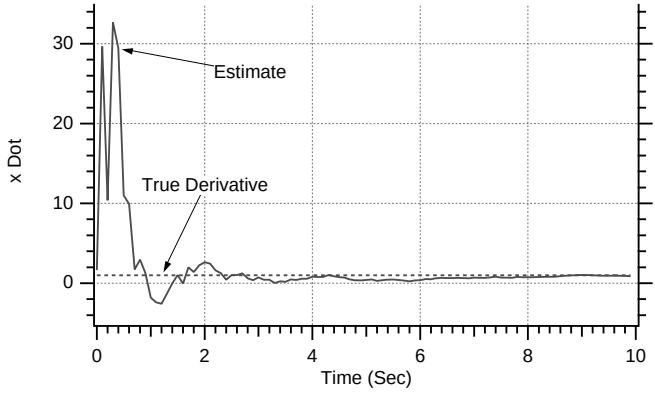

Fig. 3.9 First-order recursive least-squares filter is able to estimate derivative of signal.

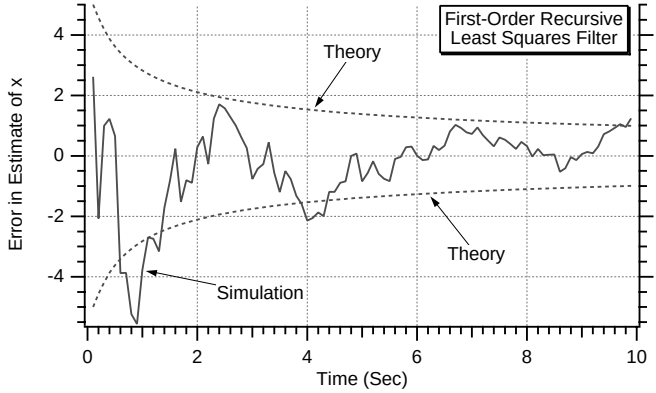

Fig. 3.10 Single-run simulation results for first state agree with theoretical formula.

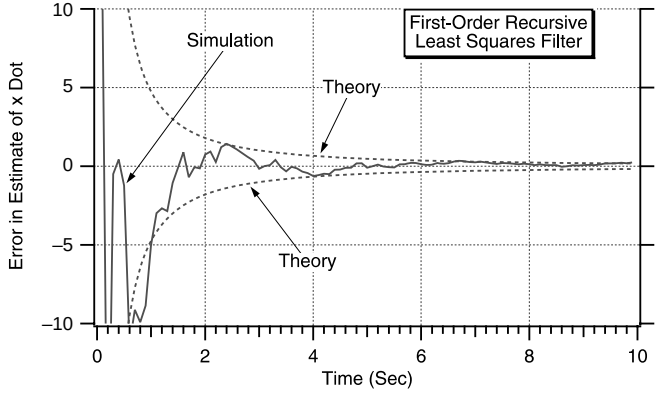

Fig. 3.11 Single-run simulation results for second state also agree with theoretical formula.

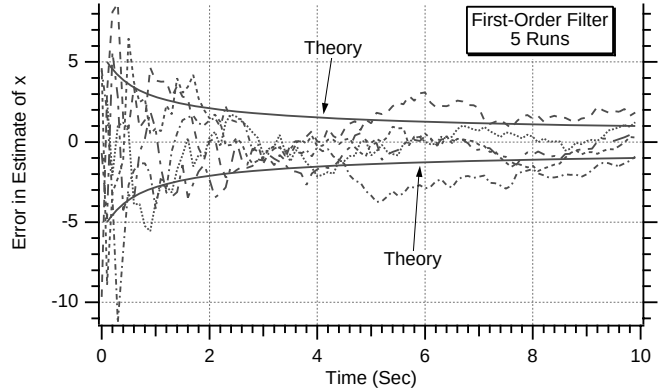

Fig. 3.12 Multiple runs indicate that simulated error in the estimates of first state appear to lie within theoretical error bounds 68% of the time.

first and second states will eventually approach zero as more measurements are taken.

If more runs were made in the case in which there was no truncation error, we would expect to find the computed error in the estimate for both states to lie between the theoretical bounds approximately 68% of the time. To ensure that the simulated and theoretical errors in the estimates agree in this way, Listing 3.3 was slightly modified so that we could run it in the Monte Carlo mode (i.e., repeated simulation trials). A five-run Monte Carlo set was run, and we can see from Figs. 3.12 and 3.13 that the simulation results on average appear to lie within the theoretical bounds approximately 68% of the time. This simple experiment confirms that the theoretical formulas presented in this section for the variance of the errors in the estimates on each of the two states are correct.

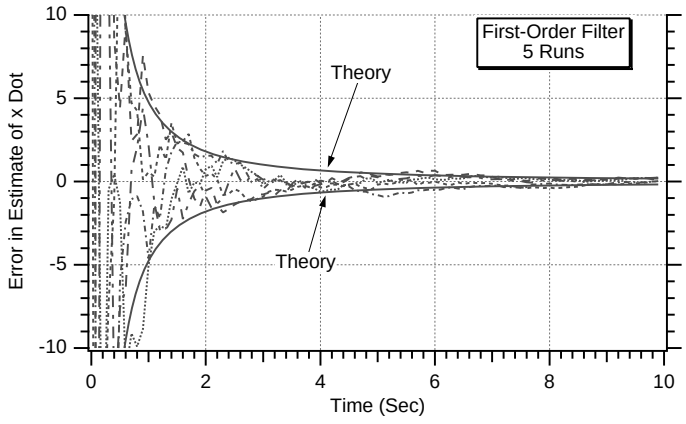

Fig. 3.13 Multiple runs indicate that simulated error in the estimates of second state appear to lie within theoretical error bounds 68% of the time.

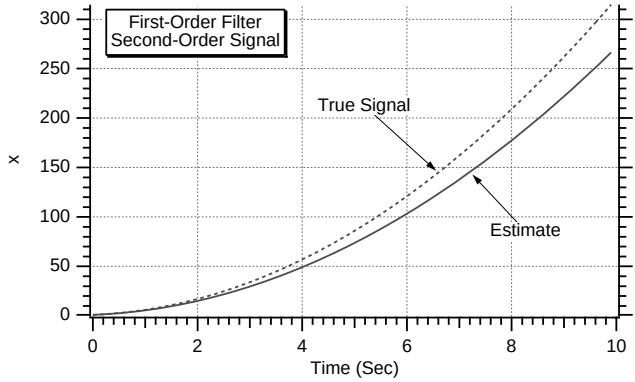

Fig. 3.14 First-order recursive least-squares filter is unable to track the first state of a second-order polynomial.

Another case was run with Listing 3.3 to examine the effect of truncation error. In this example the actual signal is now second order without noise (i.e., $A0 = 1$, $A1 = 2$, $A2 = 3$, $SIGNOISE = 0$), whereas the recursive least-squares filter remains first order. We can see from Figs. 3.14 and 3.15 that it is clear from the two filter estimates that the first-order filter cannot track the first or second state of the signal.

Figures 3.16 and 3.17 display the actual errors in the estimates of the first and second states, respectively (i.e., XHERR and XDHERR in Listing 3.3), obtained from running Listing 3.3. Also superimposed on the figures are the theoretical truncation error formulas already presented in this section (i.e., EPS and EPSD in Listing 3.3). We can see from both figures that the theoretical truncation error formulas and simulation results are in excellent agreement. Therefore, the truncation error formulas for the first-order filter are empirically validated.

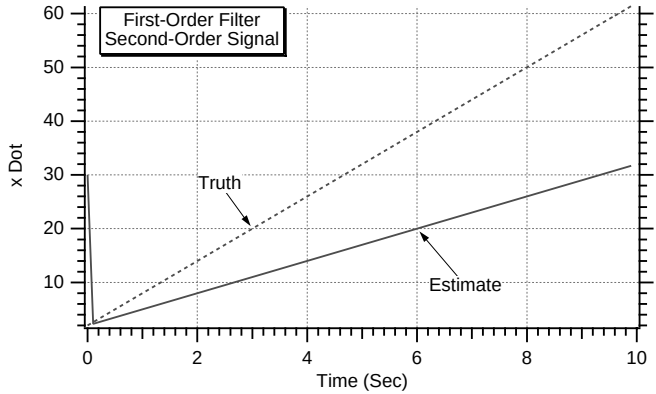

Fig. 3.15 First-order recursive least-squares filter is unable to track the second state of a second-order polynomial.

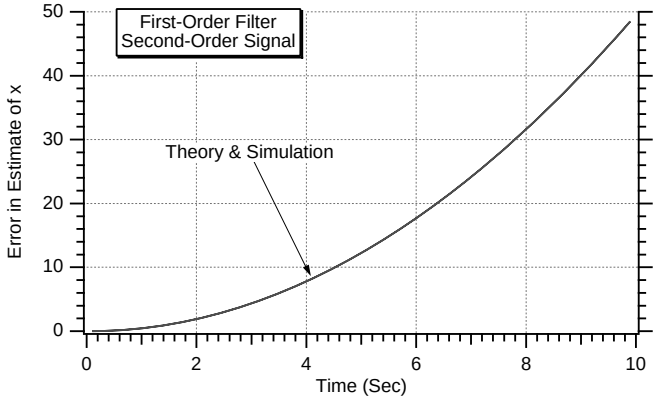

Fig. 3.16 Simulation results and truncation error for first state are in excellent agreement.

Properties of Second-Order or Three-State Filter

Using the same techniques used in the section on the recursive zeroth-order least-squares filter, the three gains of a second-order or three-state recursive least-squares filter can be shown to be[2]

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

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

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

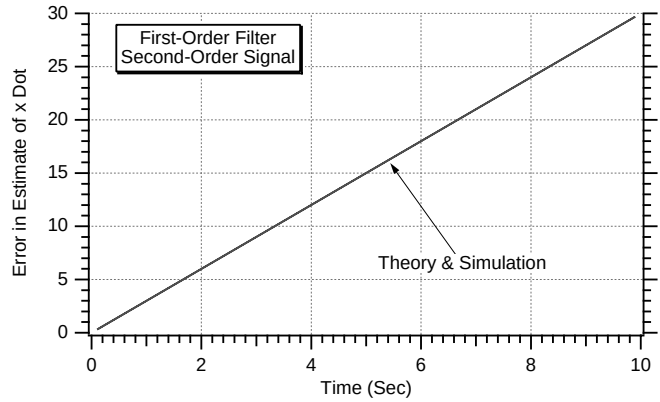

Fig. 3.17 Simulation results and truncation error for second state are in excellent agreement.

Again, we can see that all of the gains eventually go to zero as more measurements are taken (i.e., k gets larger). If we think of the measurement as a position, then the estimates from this three-state filter will be position, velocity, and acceleration (i.e., derivative of position and second derivative of position). For this filter a residual is also formed, which is the difference between the present measurement and a projection of the preceding estimate to the current time or

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

The new filter estimates are a combination of the preceding state estimates projected forward to the current time plus a gain multiplied by the residual or

$$\begin{aligned}
\hat{x}_k &= \hat{x}_{k-1} + \hat{\dot{x}}_{k-1} T_s + 0.5\hat{\ddot{x}}_{k-1} T_s^2 + K_{1_k}\text{Res}_k \\
\hat{\dot{x}}_k &= \hat{\dot{x}}_{k-1} + \hat{\ddot{x}}_{k-1} T_s^2 + K_{2_k}\text{Res}_k \\
\hat{\ddot{x}}_k &= \hat{\ddot{x}}_{k-1} + K_{3_k}\text{Res}_k
\end{aligned}$$

To understand better how these formulas apply, let us again reconsider the four measurement examples of the preceding chapter and the preceding sections where the four measurements are

$$\begin{aligned}
x_1^* &= 1.2 \\
x_2^* &= 0.2 \\
x_3^* &= 2.9 \\
x_4^* &= 2.1
\end{aligned}$$

Using the gain formulas from this section, the first set of gains for $k = 1$ can be calculated as

$$\begin{aligned}
K_{1_1} &= \frac{3(3k^2 - 3k + 2)}{k(k+1)(k+2)} = \frac{3(3*1 - 3*1 + 2)}{1(2)(3)} = 1 \\
K_{2_1} &= \frac{18(2k-1)}{k(k+1)(k+2)T_s} = \frac{18(2-1)}{1(2)(3)(1)} = 3 \\
K_{3_1} &= \frac{60}{k(k+1)(k+2)T_s^2} = \frac{60}{1(2)(3)(1)} = 10
\end{aligned}$$

Let us assume that the initial state estimates of the filter are zero or

$$\begin{aligned}
\hat{x}_0 &= 0 \\
\hat{\dot{x}}_0 &= 0 \\
\hat{\ddot{x}}_0 &= 0
\end{aligned}$$

We can now calculate the residual as

$$\text{Res}_1 = x_1^* - \hat{x}_0 - \hat{\dot{x}}_0 T_s - 0.5\hat{\ddot{x}}_0 T_s^2 = 1.2 - 0 - 0 - 0 = 1.2$$

Therefore, the new state estimates become

$$\hat{x}_1 = \hat{x}_0 + \hat{\dot{x}}_0 T_s + 0.5\hat{\ddot{x}}_0 T_s^2 + K_{1_1}\text{Res}_1 = 0 + 0 + 0 + 1*1.2 = 1.2$$
$$\hat{\dot{x}}_1 = \hat{\dot{x}}_0 + \hat{\ddot{x}}_0 T_s + K_{2_1}\text{Res}_1 = 0 + 0 + 3*1.2 = 3.6$$
$$\hat{\ddot{x}}_1 = \hat{\ddot{x}}_0 + K_{3_1}\text{Res}_1 = 0 + 10*1.2 = 12$$

Repeating the calculations for $k = 2$ yields the next set of gains:

$$K_{1_2} = \frac{3(3k^2 - 3k + 2)}{k(k+1)(k+2)} = \frac{3(3*4 - 3*2 + 2)}{2(3)(4)} = 1$$
$$K_{2_2} = \frac{18(2k-1)}{k(k+1)(k+2)T_s} = \frac{18(2*2-1)}{2(3)(4)(1)} = 2.25$$
$$K_{3_2} = \frac{60}{k(k+1)(k+2)T_s^2} = \frac{60}{2(3)(4)(1)} = 2.5$$

The next residual is based on the preceding state estimates and can be calculated as

$$\text{Res}_2 = x_2^* - \hat{x}_1 - \hat{\dot{x}}_1 T_s - 0.5\hat{\ddot{x}}_1 T_s^2 = 0.2 - 1.2 - 3.6 - 0.5*12 = -10.6$$

whereas the new state estimates become

$$\hat{x}_2 = \hat{x}_1 + \hat{\dot{x}}_1 T_s + 0.5\hat{\ddot{x}}_1 T_s^2 + K_{1_2}\text{Res}_2 = 1.2 + 3.6 + 0.5*12 + 1*(-10.6) = 0.2$$
$$\hat{\dot{x}}_2 = \hat{\dot{x}}_1 + \hat{\ddot{x}}_1 T_s + K_{2_2}\text{Res}_2 = 3.6 + 12 + 2.25*(-10.6) = -8.25$$
$$\hat{\ddot{x}}_2 = \hat{\ddot{x}}_1 + K_{3_2}\text{Res}_2 = 12 + 2.5*(-10.6) = -14.5$$

The remaining calculations for processing the third measurement are

$$K_{1_3} = \frac{3(3k^2 - 3k + 2)}{k(k+1)(k+2)} = \frac{3(3*9 - 3*3 + 2)}{3(4)(5)} = 1$$

$$K_{2_3} = \frac{18(2k-1)}{k(k+1)(k+2)T_s} = \frac{18(2*3-1)}{3(4)(5)(1)} = 1.5$$

$$K_{3_3} = \frac{60}{k(k+1)(k+2)T_s^2} = \frac{60}{3(4)(5)(1)} = 1$$

$$\begin{aligned}\text{Res}_3 &= x_3^* - \hat{x}_2 - \hat{\dot{x}}_2 T_s - 0.5\hat{\ddot{x}}_2 T_s^2 \\ &= 2.9 - 0.2 - (-8.25) - 0.5*(-14.5) = 18.2\end{aligned}$$

$$\begin{aligned}\hat{x}_3 &= \hat{x}_2 + \hat{\dot{x}}_2 T_s + 0.5\hat{\ddot{x}}_2 T_s^2 + K_{1_3}\text{Res}_3 \\ &= 0.2 - 8.25 + 0.5*(-14.5) + 1*18.2 = 2.9\end{aligned}$$

$$\hat{\dot{x}}_3 = \hat{\dot{x}}_2 + \hat{\ddot{x}}_2 T_s + K_{2_3}\text{Res}_3 = -8.25 - 14.5 + 1.5*18.2 = 4.55$$

$$\hat{\ddot{x}}_3 = \hat{\ddot{x}}_2 + K_{3_3}\text{Res}_3 = -14.5 + 1*18.2 = 3.7$$

and, finally, the calculations for the fourth measurement are given by

$$K_{1_4} = \frac{3(3k^2 - 3k + 2)}{k(k+1)(k+2)} = \frac{3(3*16 - 3*4 + 2)}{4(5)(6)} = \frac{19}{20}$$

$$K_{2_4} = \frac{18(2k-1)}{k(k+1)(k+2)T_s} = \frac{18(2*4-1)}{4(5)(6)(1)} = \frac{21}{20}$$

$$K_{3_4} = \frac{60}{k(k+1)(k+2)T_s^2} = \frac{60}{4(5)(6)(1)} = 0.5$$

$$\text{Res}_4 = x_4^* - \hat{x}_3 - \hat{\dot{x}}_3 T_s - 0.5\hat{\ddot{x}}_3 T_s^2 = 2.1 - 2.9 - 4.55 - 0.5*3.7 = -7.2$$

$$\begin{aligned}\hat{x}_4 &= \hat{x}_3 + \hat{\dot{x}}_3 T_s + 0.5\hat{\ddot{x}}_3 T_s^2 + K_{1_4}\text{Res}_4 \\ &= 2.9 + 4.55 + 0.5*3.7 + \frac{19}{20}*(-7.2) = 2.46\end{aligned}$$

$$\hat{\dot{x}}_4 = \hat{\dot{x}}_3 + \hat{\ddot{x}}_3 T_s + K_{2_4}\text{Res}_4 = 4.55 + 3.7*1 + \frac{21}{20}*(-7.2) = 0.69$$

$$\hat{\ddot{x}}_4 = \hat{\ddot{x}}_3 + K_{3_4}\text{Res}_4 = 3.7 + 0.5*(-7.2) = 0.1$$

Notice that the last estimate of the second-order recursive least-squares filter (i.e., $\hat{x}_4 = 2.46$) is again identical to the estimate obtained from the second-order least squares fit of Chapter 2.

Figure 3.18 compares the estimates from the batch-processing method of least squares for the second-order system of Chapter 2 and the second-order recursive least-squares filter of this section. Again we see from Fig. 3.18 that after all of the measurements are taken both the batch-processing and recursive least-squares filters yield identical answers. Of course, the major difference between the two techniques is that the batch-processing method requires that all of the measurements must first be taken before any estimates can be made, whereas the recursive

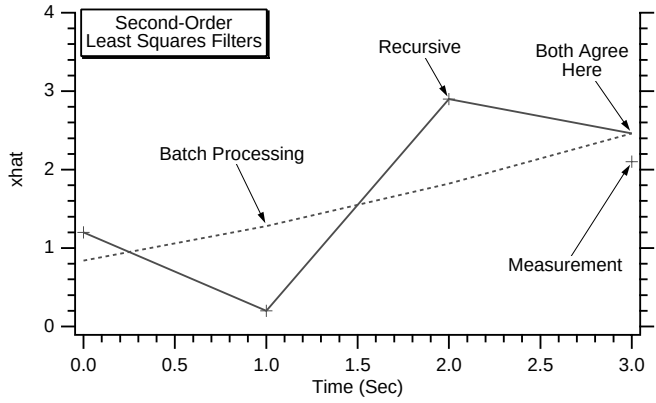

Fig. 3.18 Recursive and batch-processing second-order least-squares filters yield the same answers after all of the measurements are taken.

filter provides estimates after each measurement. In this example the recursive filter estimates pass through the first three measurements.

If the measurement data are a second-order polynomial corrupted by noise and the measurement noise statistics are known, then formulas can be derived, using techniques similar to those used on the zeroth-order filter, describing the variance of the error in the estimates of all three states (i.e., position, velocity, and acceleration) of the second-order filter. The variance of the error in the estimate for the first, second, and third states can be shown to be[2]

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

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

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

where σ_n^2 is the variance of the measurement noise and k is the number of measurements taken (i.e., $k = 1, 2, \ldots, n$). Again the variances of the errors in the estimates of all of the states decrease with increasing k. If the real signal is a third-order polynomial (one degree higher than the second-order filter) or

$$x_k^* = a_0 + a_1 t + a_2 t^2 + a_3 t^3 = a_0 + a_1(k-1)T_s + a_2(k-1)^2 T_s^2 + a_3(k-1)^3 T_s^3$$

the filter will not be able to track the signal as we saw in the preceding section. As was already mentioned, the resulting error buildup caused by this lack of tracking ability is known as truncation error ε. The truncation errors for the errors in the

estimates of the first, second, and third states of the second-order recursive least-squares filter are given by[2]

$$\varepsilon_k = \frac{1}{20} a_3 T_s^3 (k-1)(k-2)(k-3)$$

$$\dot{\varepsilon}_k = \frac{1}{10} a_3 T_s^2 (6k^2 - 15k + 11)$$

$$\ddot{\varepsilon}_k = 3a_3 T_s (k-1)$$

Listing 3.4 shows how the second-order recursive least-squares filter, along with the theoretical formulas, was programmed. We can see that measurements of a second-order signal X plus noise XS is taken every 0.1 s for 10 s. From Listing 3.4 we can tell that the actual error in the estimate XHERR in the first state is compared to plus and minus the square root of P_{11}, the actual error in the estimate of the second state XDHERR is compared to plus and minus the square root of P_{22}, and the actual error in the estimate of the third state XDDHERR is compared to plus and minus the square root of P_{33}. Nominally the simulation is set so that the true signal is also a second-order polynomial with values identical to those of the example in Chapter 2 (i.e., A0 = 2, A1 = − 2, and A2 = 5). The measurement noise has a standard deviation of 50 (SIGNOISE = 50).

Listing 3.4 Simulation for testing second-order recursive least-squares filter

```
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)
    OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
    OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
    TS=0.1
    SIGNOISE=50.
    A0=2.
    A1=-2.
    A2=5.
    A3=0.
    XH=0.
    XDH=0.
    XDDH=0.
    XN=0.
    DO 10 T=0,10.,TS
    XN=XN+1.
    CALL GAUSS(XNOISE,SIGNOISE)
    X=A0+A1*T+A2*T*T+A3*T*T*T
    XD=A1+2*A2*T+3.*A3*T*T
```

(*continued*)

Listing 3.4 (*Continued*)

```
      XDD=2*A2+6*A3*T
      XS=X+XNOISE
      XK1=3*(3*XN*XN-3*XN+2)/(XN*(XN+1)*(XN+2))
      XK2=18*(2*XN-1)/(XN*(XN+1)*(XN+2)*TS)
      XK3=60/(XN*(XN+1)*(XN+2)*TS*TS)
      RES=XS-XH-TS*XDH-0.5*TS*TS*XDDH
      XH=XH+XDH*TS+0.5*XDDH*TS*TS+XK1*RES
      XDH=XDH+XDDH*TS+XK2*RES
      XDDH=XDDH+XK3*RES
      IF(XN.EQ.1.OR.XN.EQ.2)THEN
        SP11=0
        SP22=0
        SP33=0
      ELSE
        SP11=SIGNOISE*SQRT(3*(3*XN*XN-3*XN+2)/(XN*(XN+1)*
     1    (XN+2)))
        SP22=SIGNOISE*SQRT(12*(16*XN*XN-30*XN+11)/
     1    (XN*(XN*XN-1)*(XN*XN-4)*TS*TS))
        SP33=SIGNOISE*SQRT(720/(XN*(XN*XN-1)*(XN*XN-4)
     1    *TS*TS*TS*TS))
      ENDIF
      XHERR=X-XH
      XDHERR=XD-XDH
      XDDHERR=XDD-XDDH
      EPS=A3*TS*TS*TS*(XN-1)*(XN-2)*(XN-3)/20
      EPSD=A3*TS*TS*(6*XN*XN-15*XN+11)/10
      EPSDD=3*A3*TS*(XN-1)
      WRITE(9,*)T,X,XS,XH,XD,XDH,XDD,XDDH
      WRITE(1,*)T,X,XS,XH,XD,XDH,XDD,XDDH
      WRITE(2,*)T,XHERR,SP11,-SP11,EPS,XDHERR,SP22,-SP22,EPSD,
     1   XDDHERR,SP33,-SP33,EPSDD
   10 CONTINUE
      CLOSE(1)
      CLOSE(2)
      PAUSE
      END
C SUBROUTINE GAUSS IS SHOWN IN LISTING 1.8
```

The nominal case of Listing 3.4 was run, and the true signal, measurement, and filter estimate are compared in Fig. 3.19. We can see that the actual measurement is not too noisy in this example because the amplitude of the signal is much larger than the standard deviation of the noise. We can also see that the filter estimate of the first state is excellent. Figures 3.20 and 3.21 show that after an initial transient period the second-order recursive least-squares filter is able to estimate the first and second derivatives of x quite accurately.

As was already mentioned, for more exact work the error in the estimate (i.e., difference between true signal and estimate) is used as a measure of performance. Figures 3.22–3.24 compare the error in the estimate of the first, second, and third

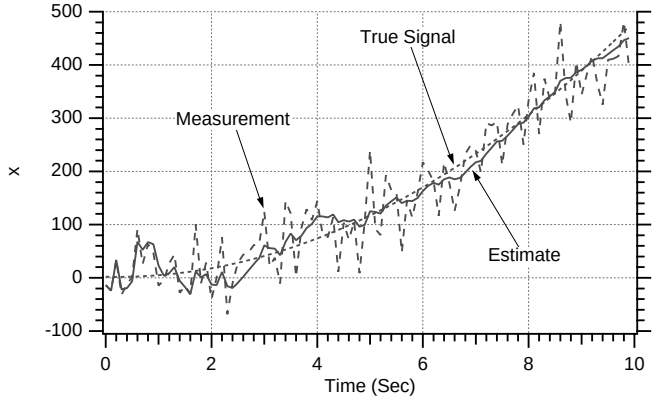

Fig. 3.19 Second-order growing memory filter is able to track second-order signal plus noise.

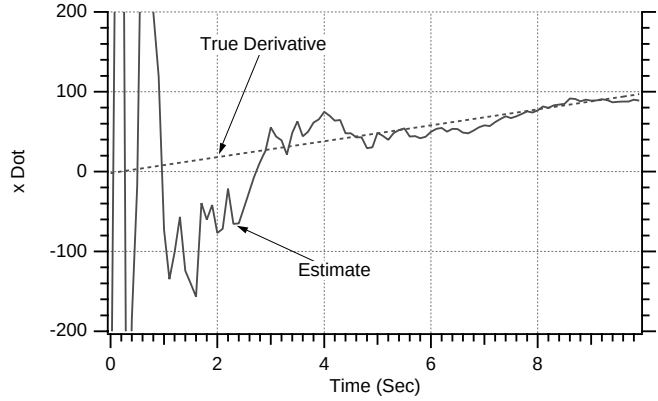

Fig. 3.20 Estimate of derivative is excellent.

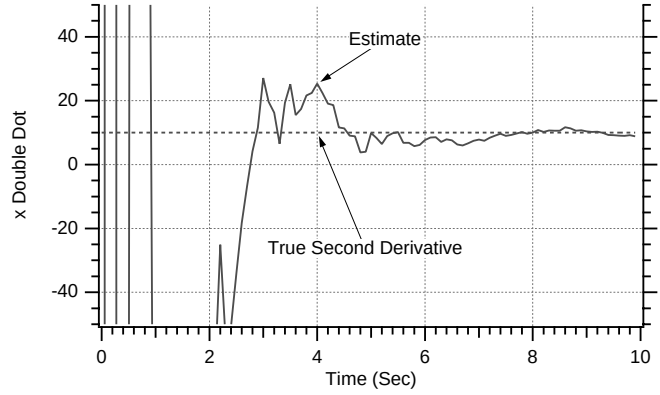

Fig. 3.21 Estimate of second derivative is also excellent.

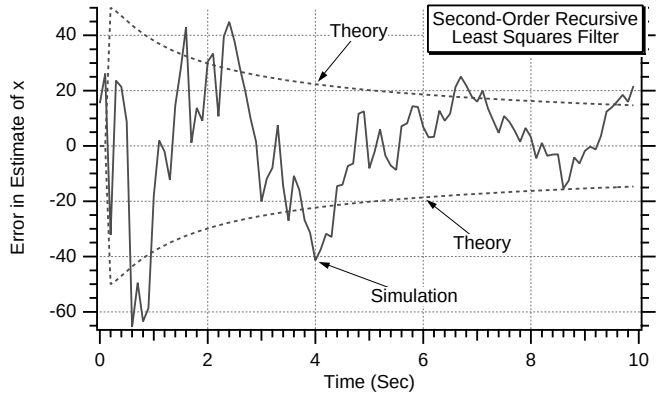

Fig. 3.22 Error in estimate of first state appears to be within theoretical error bounds.

states, respectively, to the theoretical predictions of this section (i.e., square root of P_{11}, P_{22}, and P_{33}). We can see that because the single-run simulation results are within the theoretical bounds most of the time theory and simulation appear to agree.

If more runs were made for this case, we would expect to find the computed error in the estimate for all three states to lie between the theoretical error bounds approximately 68% of the time. To ensure that the simulated and theoretical errors in the estimates agree in this way, Listing 3.4 was slightly modified so that we could run it in the Monte Carlo mode (i.e., repeated simulation trials). A five-run Monte Carlo set was run, and we can see from Figs. 3.25–3.27 that the five runs, on average, appear to lie within the theoretical error bounds approximately

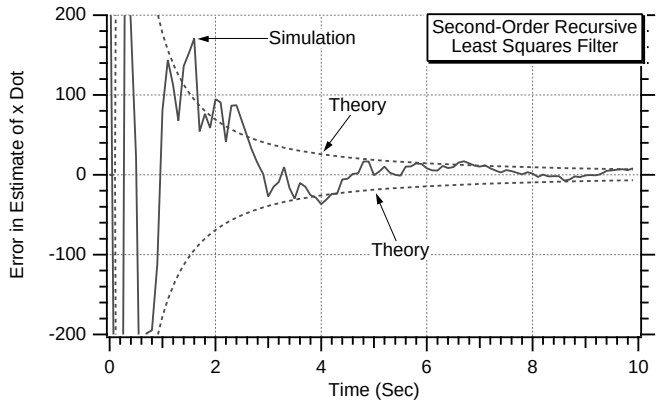

Fig. 3.23 Error in estimate of second state appears to be within theoretical error bounds.

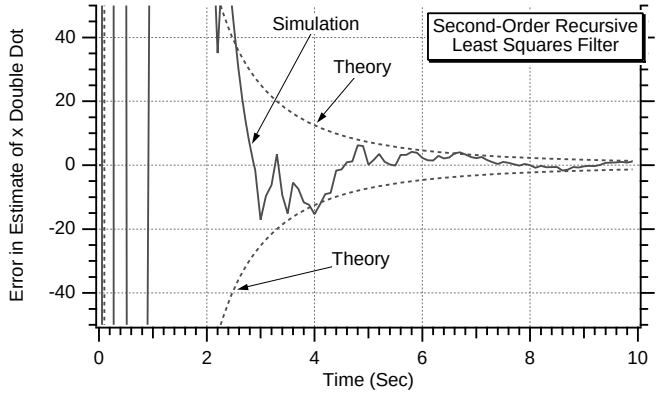

Fig. 3.24 Error in estimate of third state appears to be within theoretical error bounds.

68% of the time. This simple experiment confirms that the theoretical formulas presented in this section for the variance of the errors in the estimates on each of the three states appear to be correct.

Another case was run with Listing 3.4 to examine the effect of truncation error. In this example the actual signal is now a noise-free third-order polynomial (i.e., $A0 = 1,\ A1 = 2,\ A2 = 3,\ A3 = 4,\ \text{SIGNOISE} = 0$), whereas the recursive least-squares filter remains second order. We can see from Figs. 3.28–3.30 that it is clear that the second-order filter cannot track the first, second, or third state of the third-order signal (i.e., signal or its first and second derivatives). In all cases the filter estimate is diverging from either the true signal or the true state.

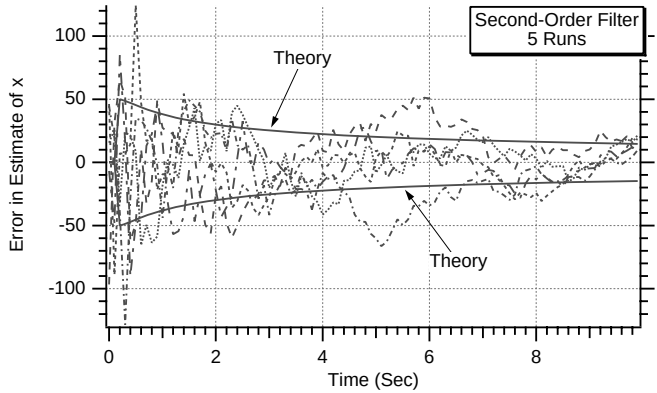

Fig. 3.25 Multiple runs indicate that on average the error in the estimate of first state appears to be within error bounds 68% of the time.

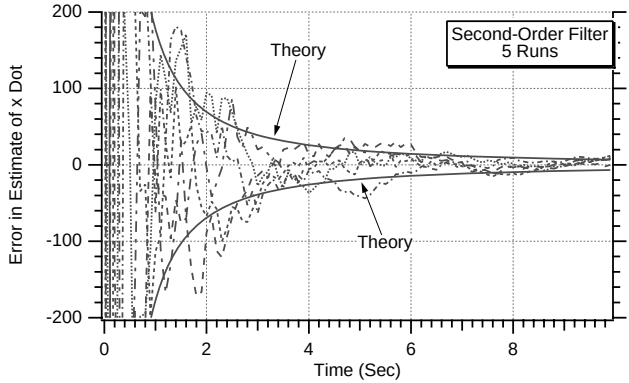

Fig. 3.26 Multiple runs indicate that on average the error in the estimate of second state appears to be within error bounds 68% of the time.

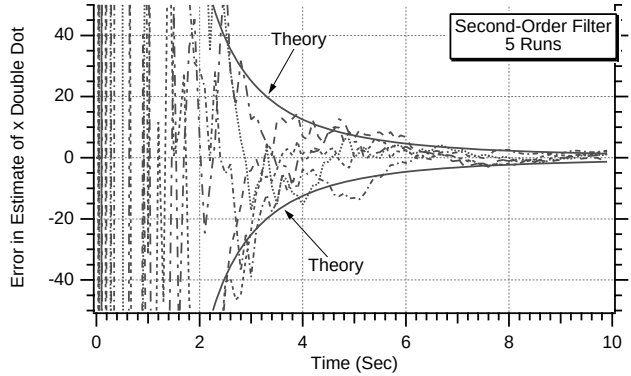

Fig. 3.27 Multiple runs indicate that on average the error in the estimate of third state appears to be within error bounds 68% of the time.

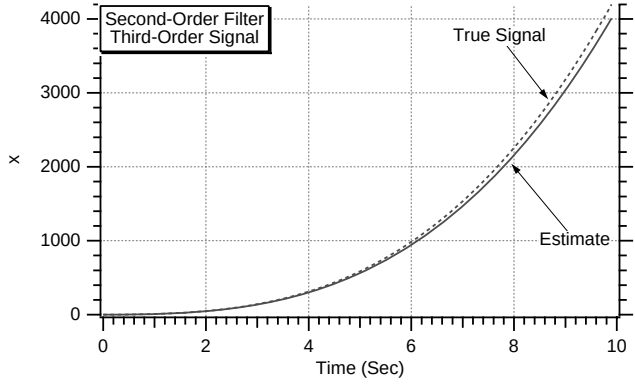

Fig. 3.28 Second-order recursive least-squares filter is unable to track the first state of a third-order polynomial.

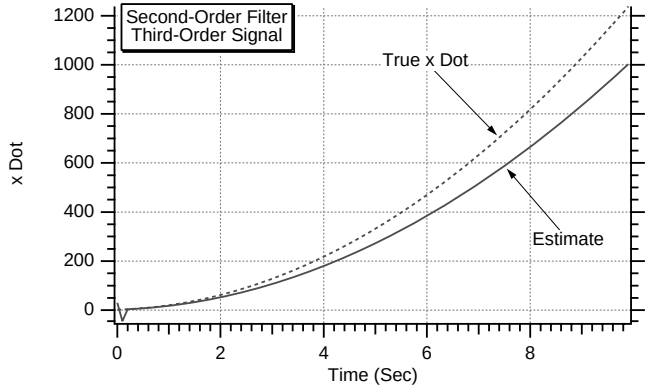

Fig. 3.29 Second-order recursive least-squares filter is unable to track the second state of a third-order polynomial.

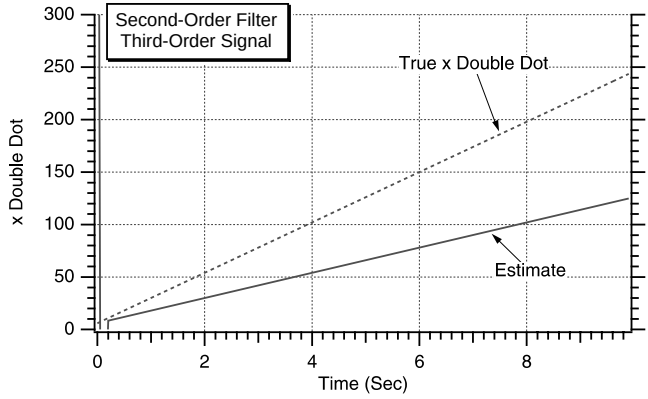

Fig. 3.30 Second-order recursive least-squares filter is unable to track the third state of a third-order polynomial.

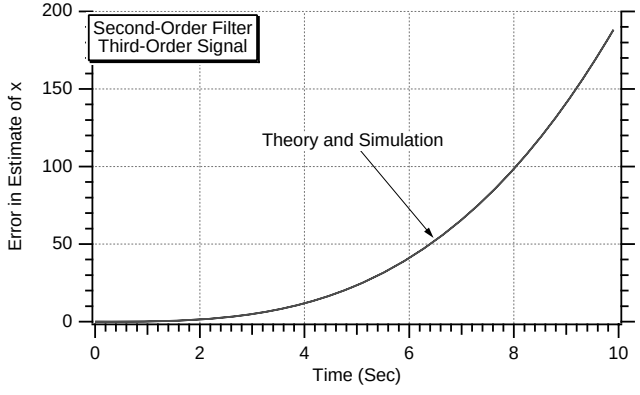

Fig. 3.31 Simulation results and truncation error formula for the first state are in excellent agreement.

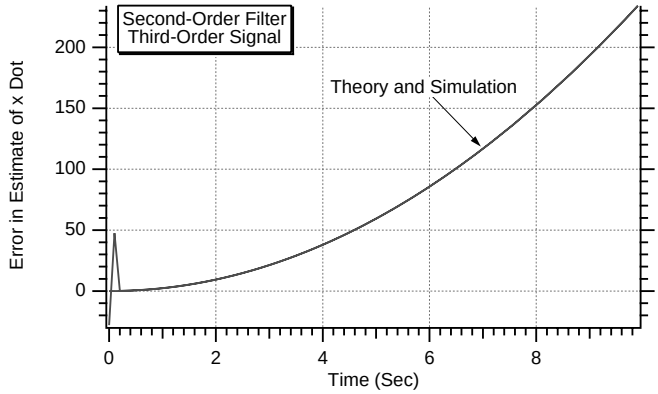

Fig. 3.32 Simulation results and truncation error formula for the second state are in excellent agreement.

Figures 3.31–3.33 display the actual errors in the estimates of the first, second, and third states, respectively, obtained from running Listing 3.4. Also superimposed on the figures are the theoretical truncation error formulas already presented in this section. We can see from the three figures that the theoretical truncation error formulas and simulation results are in excellent agreement. Therefore, the truncation error formulas for the second-order recursive least-squares filter can also be considered empirically validated.

Summary

In this chapter we have first shown that the batch-processing method of least squares could be made recursive. Table 3.3 presents the equations for the one-,

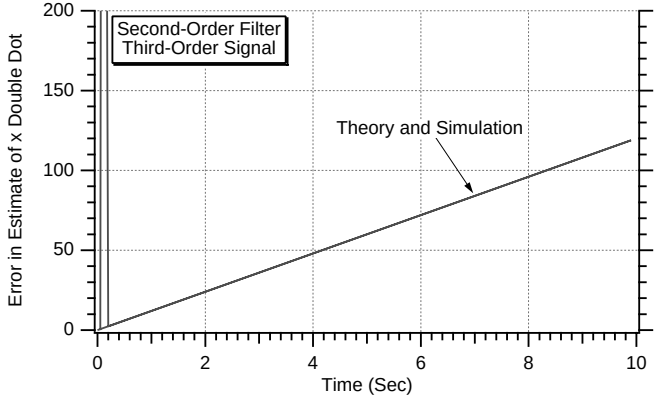

Fig. 3.33 Simulation results and truncation error formula for the third state are in excellent agreement.

two-, and three-state (i.e., also known as zeroth-, first-, and second-order) recursive least-squares filters. We can see from the second column of Table 3.3 that all three filters have the same structure in that the new state estimate is always the old state estimate projected forward to the current time plus a gain times a residual. The gain formulas for all three filters are summarized in the third column of Table 3.3.

Table 3.4 presents important theoretical performance formulas for the one-, two-, and three-state (i.e., also known as zeroth-, first-, and second-order) recursive least-squares filters. The second column represents the standard deviation of the error in the estimate for any state for each of the three filters caused by measurement noise. The third column represents the error in the estimate for any state because the measurement signal may be of higher order than the filter. All of the formulas presented in Table 3.4 were empirically validated by simulation experiment.

The formulas of Table 3.4 for the standard deviation of the errors in the estimates of the three different recursive least-squares filters were programmed so that they could be compared on a normalized basis. Figures 3.34–3.36 present normalized plots showing how the errors in the state estimates caused by measurement noise vary with the number of measurements taken and with the order of the filter used. We can see from Figs. 3.34 and 3.35 that, from a measurement noise reduction point of view, it is best to use as low an order filter as possible because that will tend to reduce the error in the estimate caused by measurement noise. However, we also know from Table 3.4 that if we make the

Table 3.3 Recursive least-square filter comparison

State	Filter	Gains
1	$\text{Res}_k = x_k^* - \hat{x}_{k-1}$ $\hat{x}_k = \hat{x}_{k-1} + K_{1_k}\text{Res}_k$	$K_{1_k} = \dfrac{1}{k}$
2	$\text{Res}_k = x_k^* - \hat{x}_{k-1} - \hat{\dot{x}}_{k-1}T_s$ $\hat{x}_k = \hat{x}_{k-1} + \hat{\dot{x}}_{k-1}T_s + K_{1_k}\text{Res}_k$ $\hat{\dot{x}}_k = \hat{\dot{x}}_{k-1} + K_{2_k}\text{Res}_k$	$K_{1_k} = \dfrac{2(2k-1)}{k(k+1)}$ $K_{2_k} = \dfrac{6}{k(k+1)T_s}$
3	$\text{Res}_k = x_k^* - \hat{x}_{k-1} - \hat{\dot{x}}_{k-1}T_s - 0.5\hat{\ddot{x}}_{k-1}T_s^2$ $\hat{x}_k = \hat{x}_{k-1} + \hat{\dot{x}}_{k-1}T_s + 0.5\hat{\ddot{x}}_{k-1}T_s^2 + K_{1_k}\text{Res}_k$ $\hat{\dot{x}}_k = \hat{\dot{x}}_{k-1} + \hat{\ddot{x}}_{k-1}T_s + K_{2_k}\text{Res}_k$ $\hat{\ddot{x}}_k = \hat{\ddot{x}}_{k-1} + K_{3_k}\text{Res}_k$	$K_{1_k} = \dfrac{3(3k^2-3k+2)}{k(k+1)(k+2)}$ $K_{2_k} = \dfrac{18(2k-1)}{k(k+1)(k+2)T_s}$ $K_{3_k} = \dfrac{60}{k(k+1)(k+2)T_s^2}$

Table 3.4 Standard deviation of errors in estimates and truncation error formulas for various order recursive least-squares filters

State	Standard deviation	Truncation error
1	$\sqrt{P_k} = \dfrac{\sigma_n}{\sqrt{k}}$	$\varepsilon_k = \dfrac{a_1 T_s}{2}(k-1)$
2	$\sqrt{P_{11_k}} = \sigma_n\sqrt{\dfrac{2(2k-1)}{k(k+1)}}$ $\sqrt{P_{22_k}} = \dfrac{\sigma_n}{T_s}\sqrt{\dfrac{12}{k(k^2-1)}}$	$\varepsilon_k = \dfrac{1}{6}a_2 T_s^2(k-1)(k-2)$ $\dot{\varepsilon}_k = a_2 T_s(k-1)$
3	$\sqrt{P_{11_k}} = \sigma_n\sqrt{\dfrac{3(3k^2-3k+2)}{k(k+1)(k+2)}}$ $\sqrt{P_{22_k}} = \dfrac{\sigma_n}{T_s}\sqrt{\dfrac{12(16k^2-30k+11)}{k(k^2-1)(k^2-4)}}$ $\sqrt{P_{33_k}} = \dfrac{\sigma_n}{T_s^2}\sqrt{\dfrac{720}{k(k^2-1)(k^2-4)}}$	$\varepsilon_k = \dfrac{1}{20}a_3 T_s^3(k-1)(k-2)(k-3)$ $\dot{\varepsilon}_k = \dfrac{1}{10}a_3 T_s^2(6k^2-15k+11)$ $\ddot{\varepsilon}_k = 3a_3 T_s(k-1)$

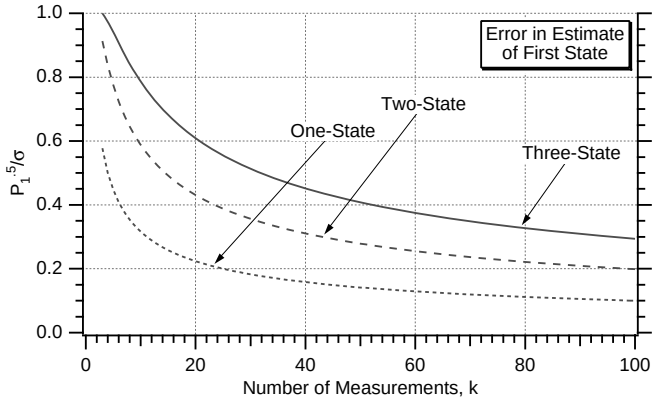

Fig. 3.34 Error in the estimate of the first state decreases with decreasing filter order and increasing number of measurements taken.

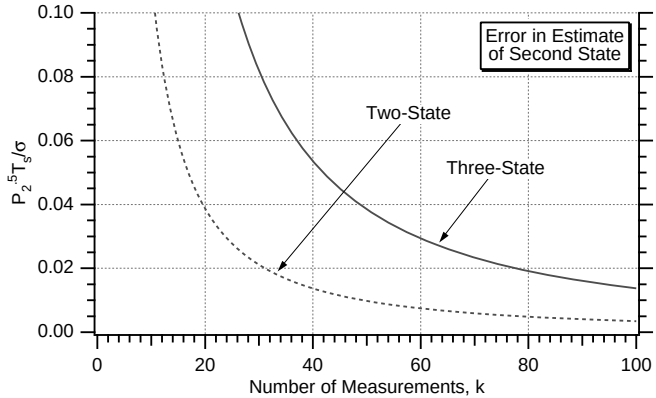

Fig. 3.35 Error in the estimate of the second state decreases with decreasing filter order and increasing number of measurements taken.

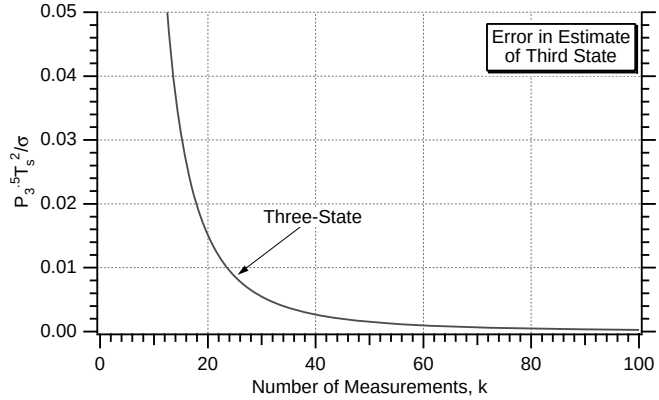

Fig. 3.36 Error in the estimate of the third state decreases with increasing number of measurements taken.

order of the filter too low there might be excessive truncation error. In each case the best-order filter to use will depend on the number of measurements to be taken and the actual order of the signal.

References

[1]Selby, S. M., *Standard Mathematical Tables*, 20th ed., Chemical Rubber Co., Cleveland, OH, 1972, p. 37.

[2]Morrison, N., *Introduction to Sequential Smoothing and Prediction*, McGraw–Hill, New York, 1969, pp. 339–376.