Method of Least Squares

Method of Least Squares
Overview

- **Deriving formulas for different least squares filters**
  - **Zeroth-order or one-state filter**
  - **First-order or two-state filter**
  - **Second-order or three-state filter**
  - **Third-Order or Four-State System**
- **Experiments with each of the filters**
  - **Signal contaminated with noise**
  - **Look at estimates and errors in the estimates**
- **Filter comparison**
- **Accelerometer testing example**

What We Are Going To Do

- Assume a polynomial form to represent signal
- Estimate the coefficients of the selected polynomial by choosing a goodness of fit criterion
- Use calculus to minimize the sum of the squares of the individual discrepancies in order to obtain the best coefficients for the selected polynomial

Zeroth-Order or One-State Filter

Least Squares Method For Zeroth-Order System-1

We want to minimize

$$R = \sum_{k=1}^{n} (\hat{x}_k - x_k^*)^2 = \sum_{k=1}^{n} (a_0 - x_k^*)^2$$

Estimates ($\hat{x}_k$)

Measurements (x_k^*)

Expansion yields

$$R = \sum_{k=1}^{n} (\hat{x}_k - x_k^*)^2 = (a_0 - x_1^*)^2 + (a_0 - x_2^*)^2 + \ldots + (a_0 - x_n^*)^2$$

Using calculus we can minimize R

$$\frac{\partial R}{\partial a_0} = 0 = 2(a_0 - x_1^*) + 2(a_0 - x_2^*) + \ldots + 2(a_0 - x_n^*)$$

Recognizing that

$$- x_1^* - x_2^* - \ldots - x_n^* = - \sum_{k=1}^{n} x_k^*$$ **and** $$a_0 + a_0 + \ldots + a_0 = na_0$$

Least Squares Method For Zeroth-Order System-2

We get

$$0 = na_0 - \sum_{k=1}^{n} x_k^*$$

Rearranging terms yields

$$a_0 = \frac{\sum_{k=1}^{n} x_k^*}{n}$$

Since

$$\hat{x}_k = a_0$$

- **We can say that the best constant fit to a set of measurement data in the least squares sense is simply the average value of the measurements!**

- **Note that this is a batch processing technique since all the data must be collected before an estimate can be made**

Numerical Example For Zeroth-Order System

Sample measurement data

t	0	1	2	3
x*	1.2	.2	2.9	2.1

We can express time in terms of the sampling time

$t = (k-1)T_s \qquad k = 1,2,3,...$

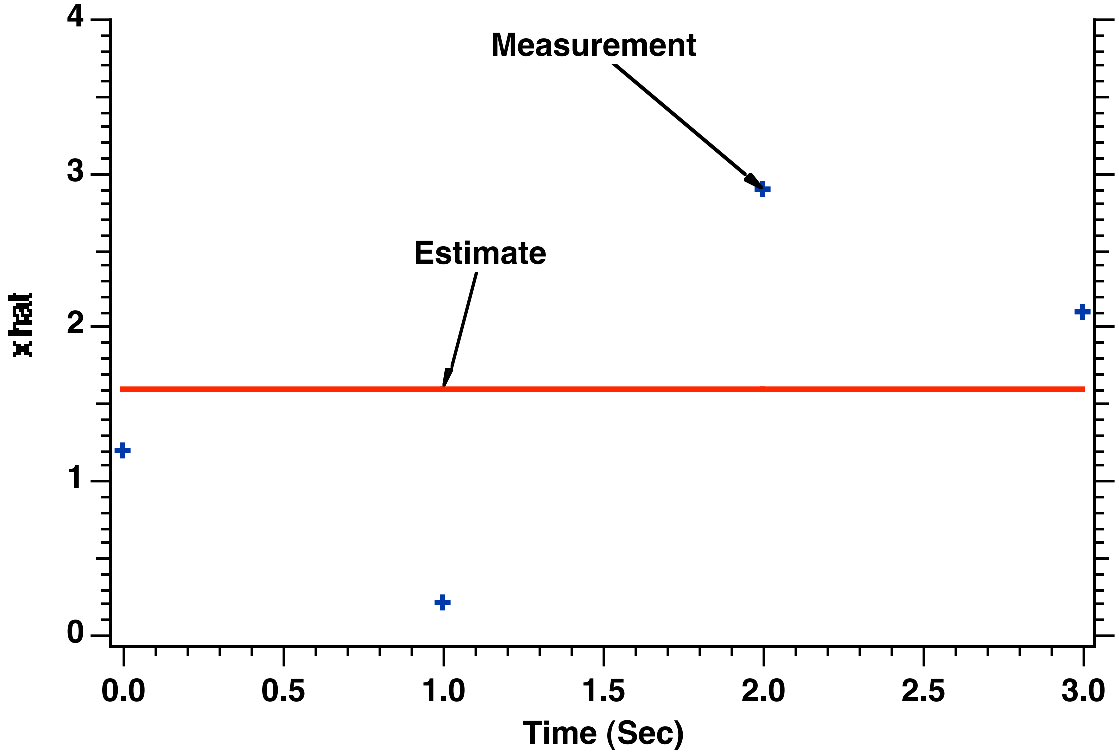
Another way of expressing the measurement data

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

Solving for the best constant yields

$$\hat{x}_k = a_0 = \frac{\sum_{k=1}^{n} x_k^*}{n} = \frac{1.2+.2+2.9+2.1}{4} = 1.6$$

The Fit To Measurement Data is Poor With a Constant of 1.6



***Can't blame method of least squares since we specified zeroth-order polynomial**

Check To See If We Minimized R

Using method of least squares value of 1.6

$$R = \sum_{k=1}^{4} (a_0 - x_k^*)^2 = (1.6 - 1.2)^2 + (1.6 - .2)^2 + (1.6 - 2.9)^2 + (1.6 - 2.1)^2 = 4.06$$

Using a larger value of 2 yields larger R

$$R = \sum_{k=1}^{4} (a_0 - x_k^*)^2 = (2 - 1.2)^2 + (2 - .2)^2 + (2 - 2.9)^2 + (2 - 2.1)^2 = 4.70$$

Using a smaller value of 1 also yields larger R

$$R = \sum_{k=1}^{4} (a_0 - x_k^*)^2 = (1 - 1.2)^2 + (1 - .2)^2 + (1 - 2.9)^2 + (1 - 2.1)^2 = 5.50$$

Therefore it appears that a constant of 1.6 minimizes R

First-Order or Two-State Filter

Least Squares Method For First-Order System-1

Fit measurement data with "best" straight line

$$\hat{x} = a_0 + a_1 t$$

Or in discrete form

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

We still want to minimize residual R

$$R = \sum_{k=1}^{n} (\hat{x}_k - x_k^*)^2 = \sum_{k=1}^{n} [a_0 + a_1(k-1)T_s - x_k^*]^2$$

We can expand R

$$R = \sum_{k=1}^{n} [a_0 + a_1(k-1)T_s - x_k^*]^2 = (a_0 - x_1^*)^2 + (a_0 + a_1 T_s - x_2^*)^2 + \ldots + (a_0 + a_1(n-1)T_s - x_n^*)^2$$

Minimize R by setting derivatives to zero

$$\frac{\partial R}{\partial a_0} = 0 = 2(a_0 - x_1^*) + 2(a_0 + a_1 T_s - x_2^*) + \ldots + 2[a_0 + a_1(n-1)T_s - x_n^*]$$

$$\frac{\partial R}{\partial a_1} = 0 = 2(a_0 + a_1 T_s - x_2^*)T_s + \ldots + 2(n-1)T_s[a_0 + a_1(n-1)T_s - x_n^*]$$

Least Squares Method For First-Order System-2

We can simplify preceding two equations

$$na_0 + a_1 \sum_{k=1}^{n} (k-1)T_s = \sum_{k=1}^{n} x_k^*$$

$$a_0 \sum_{k=1}^{n} (k-1)T_s + a_1 \sum_{k=1}^{n} [(k-1)T_s]^2 = \sum_{k=1}^{n} (k-1)T_s x_k^*$$

These equations can also be expressed in matrix form as

$$\begin{bmatrix} n & \sum_{k=1}^{n} (k-1)T_s \\ \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} ((k-1)T_s)^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n} x_k^* \\ \sum_{k=1}^{n} (k-1)T_s x_k^* \end{bmatrix}$$

We can solve for the coefficients by matrix inversion

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} (k-1)T_s \\ \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} x_k^* \\ \sum_{k=1}^{n} (k-1)T_s x_k^* \end{bmatrix}$$

Numerical Example For First-Order System

Recall

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

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n}(k-1)T_s \\ \sum_{k=1}^{n}(k-1)T_s & \sum_{k=1}^{n}[(k-1)T_s]^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} x_k^* \\ \sum_{k=1}^{n}(k-1)T_s x_k^* \end{bmatrix}$$

Intermediate calculations

$$\sum_{k=1}^{n}(k-1)T_s = 0+1+2+3=6$$

$$\sum_{k=1}^{n} x_k^* = 1.2+.2+2.9+2.1 = 6.4$$

$$\sum_{k=1}^{n}[(k-1)T_s]^2 = 0^2+1^2+2^2+3^2 = 14$$

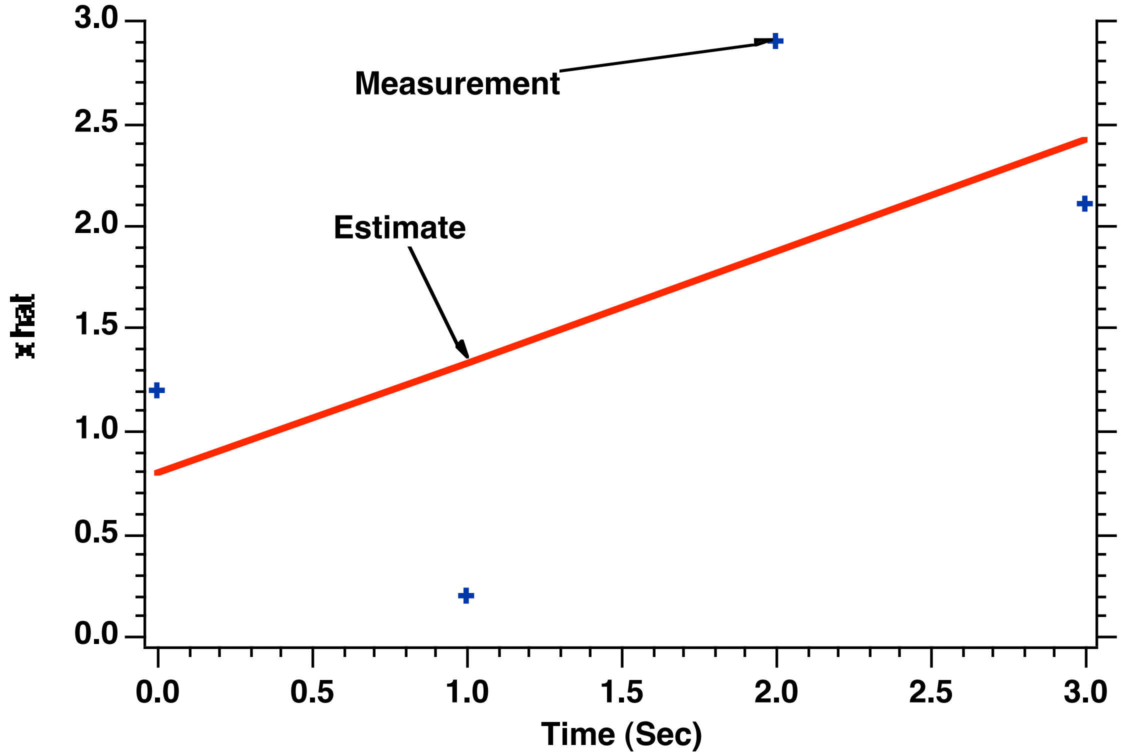
$$\sum_{k=1}^{n}(k-1)T_s x_k^* = 0*1.2+1*.2+2*2.9+3*2.1=12.3$$

***Note that this is a batch processing technique since all the data must be collected before an estimate can be made**

Therefore

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 4 & 6 \\ 6 & 14 \end{bmatrix}^{-1} \begin{bmatrix} 6.4 \\ 12.3 \end{bmatrix} = \begin{bmatrix} 7.9 \\ .54 \end{bmatrix} \longrightarrow \hat{x}_k = .79 + .54(k-1)T_s$$

Straight Line Fit to Data is Better Than Constant Fit



Straight line fit residual is smaller than constant fit residual

$$R = [.79+.54(0)-1.2]^2 + [.79+.54(1)-.2]^2 + [.79+.54(2)-2.9]^2 + [.79+.54(3)-2.1]^2 = 2.61$$

Second-Order Or Three-State Least Squares Filter

Least Squares Method For Second-Order System-1

Fit measurement data with “best” parabola

$$\hat{x} = a_0 + a_1 t + a_2 t^2$$

Or in discrete form

$$\hat{x}_k = a_0 + a_1(k-1)T_s + a_2[(k-1)T_s]^2$$

We still want to minimize residual R

$$R = \sum_{k=1}^{n} (\hat{x}_k - x_k^*)^2 = \sum_{k=1}^{n} [a_0 + a_1(k-1)T_s + a_2(k-1)^2T_s^2 - x_k^*]^2$$

We can expand R

$$R = (a_0-x_1^*)^2 + [a_0+a_1T_s+a_2T_s^2-x_2^*)]^2 + ... + [a_0+a_1(n-1)T_s+a_2(n-1)^2T_s^2-x_n^*]^2$$

Minimize R by setting derivatives to zero

$$\frac{\partial R}{\partial a_0} = 0 = 2(a_0-x_1^*) + 2[a_0+a_1T_s+a_2T_s^2-x_2^*)] + ... + 2[a_0+a_1(n-1)T_s+a_2(n-1)^2T_s^2-x_n^*]$$

$$\frac{\partial R}{\partial a_1} = 0 = 2[a_0+a_1T_s+a_2T_s^2-x_2^*)]T_s + ... + 2[a_0+a_1(n-1)T_s+a_2(n-1)^2T_s^2-x_n^*](n-1)T_s$$

$$\frac{\partial R}{\partial a_2} = 0 = 2[a_0+a_1T_s+a_2T_s^2-x_2^*)]T_s^2 + ... + 2[a_0+a_1(n-1)T_s+a_2(n-1)^2T_s^2-x_n^*](n-1)^2T_s^2$$

Least Squares Method For Second-Order System-2

We can simplify preceding three equations

$$na_0 + a_1\sum_{k=1}^{n}(k-1)T_s + a_2\sum_{k=1}^{n}[(k-1)T_s]^2 = \sum_{k=1}^{n}x_k^*$$

$$a_0\sum_{k=1}^{n}(k-1)T_s + a_1\sum_{k=1}^{n}[(k-1)T_s]^2 + a_2\sum_{k=1}^{n}[(k-1)T_s]^3 = \sum_{k=1}^{n}(k-1)T_s x_k^*$$

$$a_0\sum_{k=1}^{n}[(k-1)T_s]^2 + a_1\sum_{k=1}^{n}[(k-1)T_s]^3 + a_2\sum_{k=1}^{n}[(k-1)T_s]^4 = \sum_{k=1}^{n}[(k-1)T_s]^2 x_k^*$$

These equations can also be expressed in matrix form as

$$\begin{bmatrix} n & \sum_{k=1}^{n}(k-1)T_s & \sum_{k=1}^{n}[(k-1)T_s]^2 \\ \sum_{k=1}^{n}(k-1)T_s & \sum_{k=1}^{n}[(k-1)T_s]^2 & \sum_{k=1}^{n}[(k-1)T_s]^3 \\ \sum_{k=1}^{n}[(k-1)T_s]^2 & \sum_{k=1}^{n}[(k-1)T_s]^3 & \sum_{k=1}^{n}[(k-1)T_s]^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{k=1}^{n}x_k^* \\ \sum_{k=1}^{n}(k-1)T_s x_k^* \\ \sum_{k=1}^{n}[(k-1)T_s]^2 x_k^* \end{bmatrix}$$

Least Squares Method For Second-Order System-3

We can solve for the coefficients by matrix inversion

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 \\ \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 & \sum_{k=1}^{n} [(k-1)T_s]^3 \\ \sum_{k=1}^{n} [(k-1)T_s]^2 & \sum_{k=1}^{n} [(k-1)T_s]^3 & \sum_{k=1}^{n} [(k-1)T_s]^4 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} x_k^* \\ \sum_{k=1}^{n} (k-1)T_s x_k^* \\ \sum_{k=1}^{n} [(k-1)T_s]^2 x_k^* \end{bmatrix}$$

***Note that this is a batch processing technique since all the data must be collected before an estimate can be made**

MATLAB Program to Solve For Three Coefficients

```
T(1)=0;
T(2)=1;
T(3)=2;
T(4)=3;
X(1)=1.2;
X(2)=.2;
X(3)=2.9;
X(4)=2.1;
N=4;
SUM1=0;
SUM2=0;
SUM3=0;
SUM4=0;
SUM5=0;
SUM6=0;
SUM7=0;
for I=1:4
        SUM1=SUM1+T(I);
        SUM2=SUM2+T(I)*T(I);
        SUM3=SUM3+X(I);
        SUM4=SUM4+T(I)*X(I);
        SUM5=SUM5+T(I)*T(I)*T(I);
        SUM6=SUM6+T(I)*T(I)*T(I)*T(I);
        SUM7=SUM7+T(I)*T(I)*X(I);
end
A(1,1)=N;
A(1,2)=SUM1;
A(1,3)=SUM2;
A(2,1)=SUM1;
A(2,2)=SUM2;
A(2,3)=SUM5;
A(3,1)=SUM2;
A(3,2)=SUM5;
A(3,3)=SUM6;
AINV=inv(A);
B(1,1)=SUM3;
B(2,1)=SUM4;
B(3,1)=SUM7;
ANS=AINV*B
```

Data

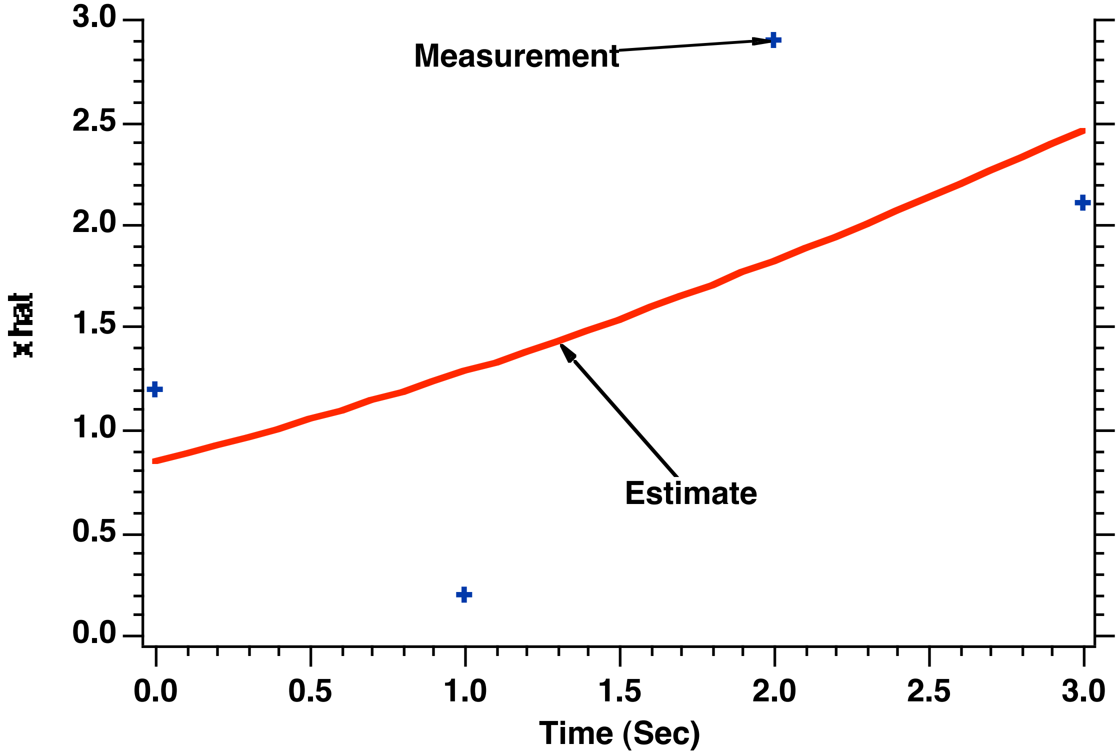
k	1	2	3	4
$(k-1)T_s$	0	1	2	3
x^*_k	1.2	.2	2.9	2.1

$$\mathbf{ANS} = \begin{bmatrix} .84 \\ .36 \\ .05 \end{bmatrix}$$

Or

$$\hat{x}_k = .84 + .39(k-1)T_s + .05[(k-1)T_s]^2$$

Parabolic Fit To Data is Pretty Good Too



Parabolic fit residual is smaller than constant or straight line fit residual

$$R = [.84+.39(0)+.05(0)-1.2]^2 + [.84+.39(1)+.05(1)-.2]^2 + [.84+.39(2)+.05(4)-2.9]^2 + [.84+.39(3)+.05(9)-2.1]^2 = 2.60$$

Least Squares Method For Third-Order System

Fit measurement data with “best” Cubic

$$\hat{x} = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

Or in discrete form

$$\hat{x}_k = a_0 + a_1 (k-1)T_s + a_2[(k-1)T_s]^2 + a_3[(k-1)T_s]^3$$

We still want to minimize residual R

$$R = \sum_{k=1}^{n} (\hat{x}_k - x_k^*)^2$$

Using same minimization techniques as before

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 & \sum_{k=1}^{n} [(k-1)T_s]^3 \\ \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 & \sum_{k=1}^{n} [(k-1)T_s]^3 & \sum_{k=1}^{n} [(k-1)T_s]^4 \\ \sum_{k=1}^{n} [(k-1)T_s]^2 & \sum_{k=1}^{n} [(k-1)T_s]^3 & \sum_{k=1}^{n} [(k-1)T_s]^4 & \sum_{k=1}^{n} [(k-1)T_s]^5 \\ \sum_{k=1}^{n} [(k-1)T_s]^3 & \sum_{k=1}^{n} [(k-1)T_s]^4 & \sum_{k=1}^{n} [(k-1)T_s]^5 & \sum_{k=1}^{n} [(k-1)T_s]^6 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} x_k^* \\ \sum_{k=1}^{n} (k-1)T_s x_k^* \\ \sum_{k=1}^{n} [(k-1)T_s]^2 x_k^* \\ \sum_{k=1}^{n} [(k-1)T_s]^3 x_k^* \end{bmatrix}$$

MATLAB Program To Solve For Four Coefficients

```
T(1)=0;
T(2)=1;
T(3)=2;
T(4)=3;
X(1)=1.2;
X(2)=.2;
X(3)=2.9;
X(4)=2.1;
N=4;
SUM1=0;
SUM2=0;
SUM3=0;
SUM4=0;
SUM5=0;
SUM6=0;
SUM7=0;
SUM8=0;
SUM9=0;
SUM10=0;
for I=1:4
        SUM1=SUM1+T(I);
        SUM2=SUM2+T(I)*T(I);
        SUM3=SUM3+X(I);
        SUM4=SUM4+T(I)*X(I);
        SUM5=SUM5+T(I)^3;
        SUM6=SUM6+T(I)^4;
        SUM7=SUM7+T(I)*T(I)*X(I);
        SUM8=SUM8+T(I)^5;
        SUM9=SUM9+T(I)^6;
        SUM10=SUM10+T(I)*T(I)*T(I)*X(I);
end
A(1,1)=N;
A(1,2)=SUM1;
A(1,3)=SUM2;
A(1,4)=SUM5;
A(2,1)=SUM1;
A(2,2)=SUM2;
A(2,3)=SUM5;
A(2,4)=SUM6;
A(3,1)=SUM2;
A(3,2)=SUM5;
A(3,3)=SUM6;
A(3,4)=SUM8;
A(4,1)=SUM5;
A(4,2)=SUM6;
A(4,3)=SUM8;
A(4,4)=SUM9;
AINV=inv(A);
B(1,1)=SUM3;
B(2,1)=SUM4;
B(3,1)=SUM7;
B(4,1)=SUM10;
ANS=AINV*B
```

Data

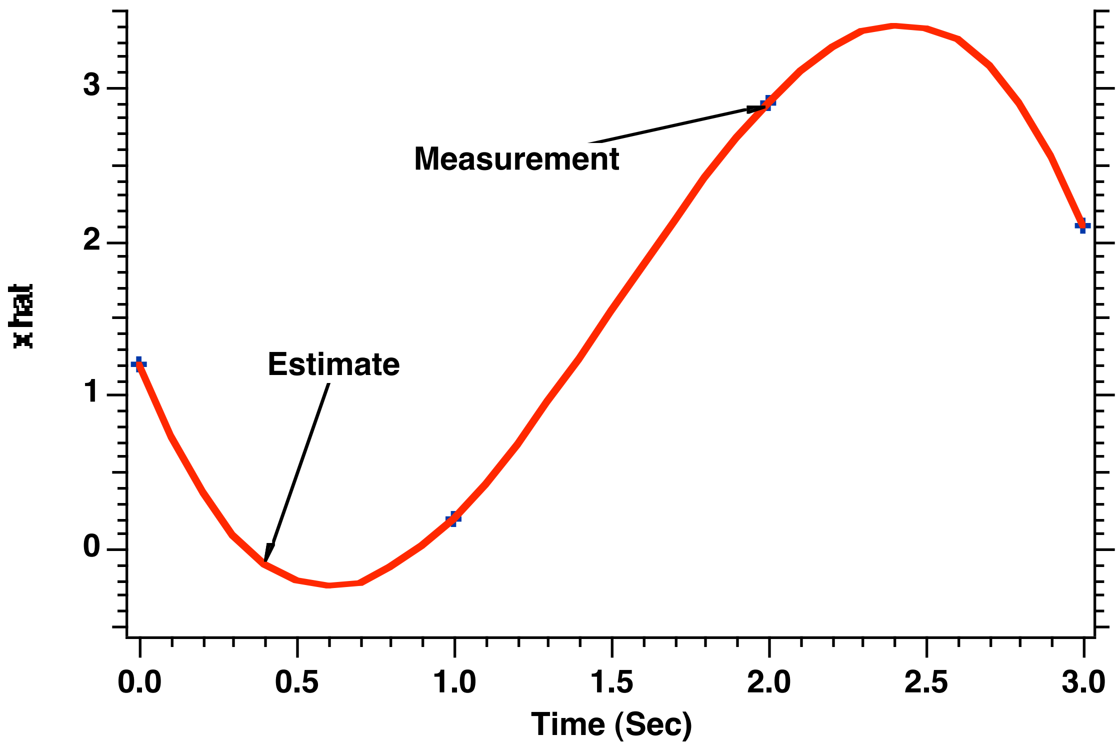
k	1	2	3	4
$(k-1)T_s$	0	1	2	3
x_k^*	1.2	.2	2.9	2.1

$$\mathbf{ANS} = \begin{bmatrix} 1.2 \\ -5.25 \\ 5.45 \\ -1.2 \end{bmatrix}$$

Or

$$\hat{x}_k = 1.2 - 5.25(k-1)T_s + 5.45[(k-1)T_s]^2 - 1.2[(k-1)T_s]^3$$

Third-Order Fit Goes Through All Four Measurements!



Cubic fit residual is zero

$$R = [1.2-5.25(0)+5.45(0)-1.2(0)-1.2]^2 + [1.2-5.25(1)+5.45(1)-1.2(1)-.2]^2$$
$$+ [1.2-5.25(2)+5.45(4)-1.2(8)-2.9]^2 + [1.2-5.25(3)+5.45(9)-1.2(27)-2.1]^2 = 0$$

For Least Squares Fit We Don't Want To Always Minimize Residual

Cases considered

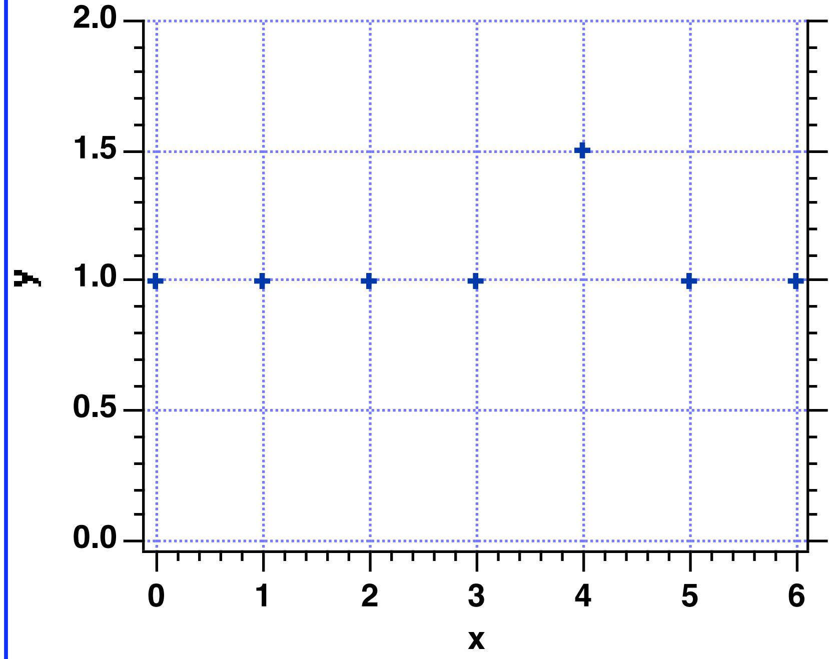
System Order	R
0	4.06
1	2.61
2	2.60
3	0

← **Estimate passes through all measurements**

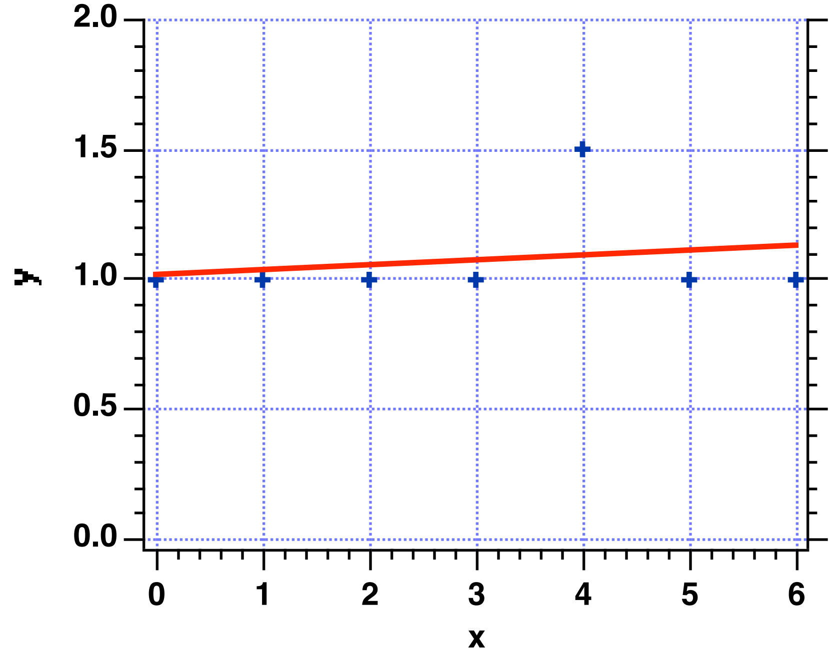
- **The residual is the difference between estimate and <u>measurement</u>**
- **Making the residual zero simply means that we pass polynomial through all the measurements**
  - **This will be bad when we consider noisy measurements**

Another Example of Fitting Data With Various Order Polynomials-1

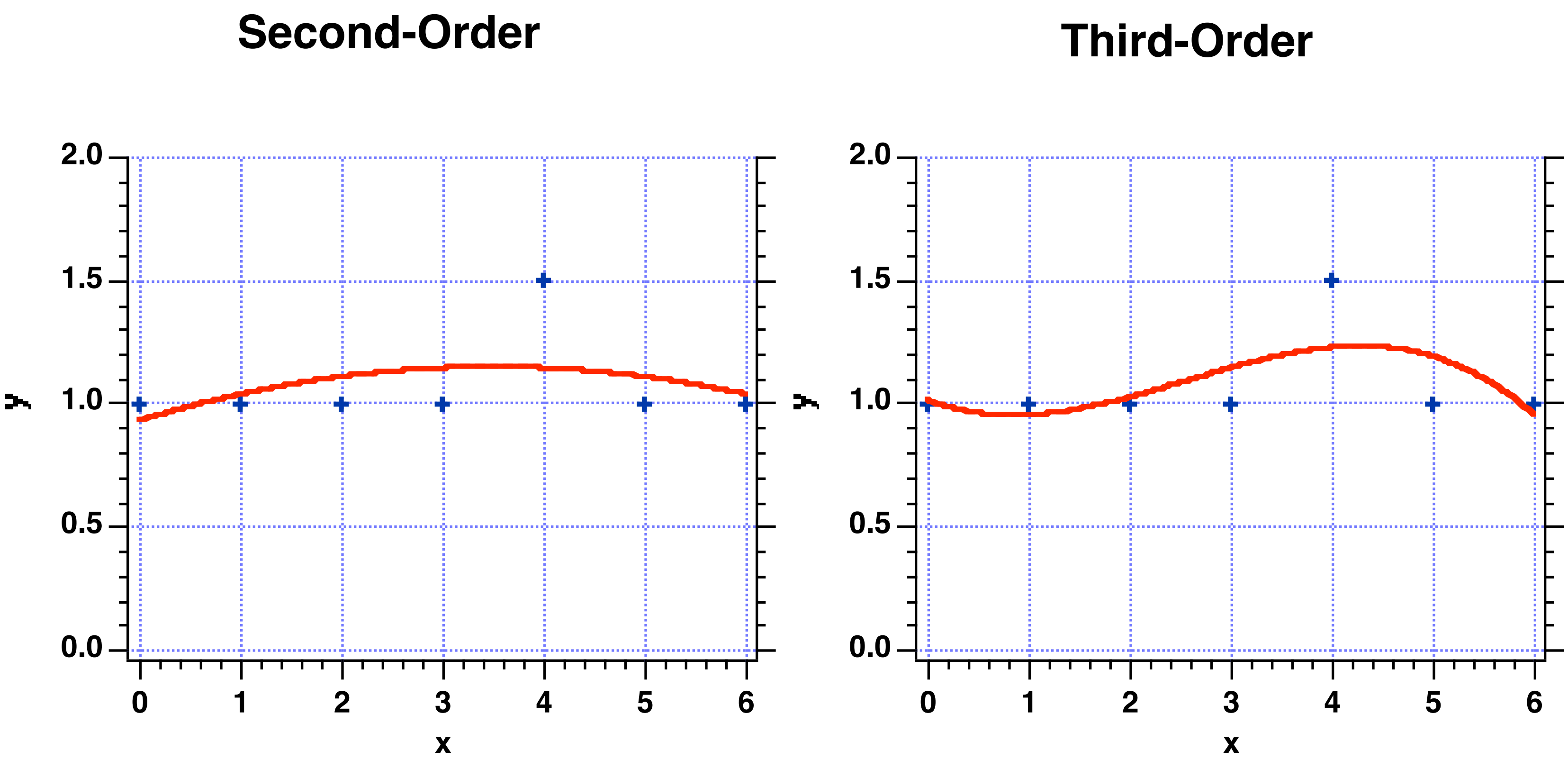
Data



First-Order



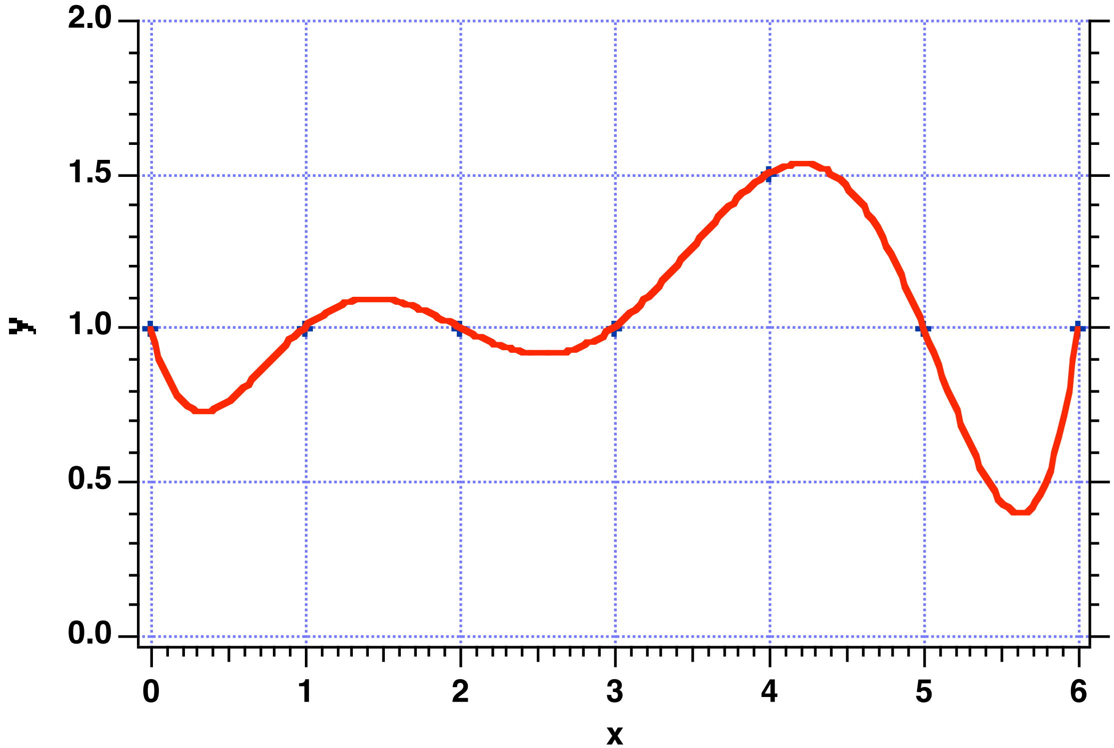
Another Example of Fitting Data With Various Order Polynomials -2



Another Example of Fitting Data With Various Order Polynomials -3

Another Example of Fitting Data With Various Order Polynomials -4

Sixth-Order



Experiments With Zeroth-Order or One-State Least Squares Filter

$$x_k^* = \text{Signal} + \text{Noise}$$

$$a_0 = \frac{\sum_{k=1}^{n} x_k^*}{n}$$

$$\hat{x}_k = a_0$$

Measurements considered

$x^* = 1 + \text{noise}$, $\sigma_{noise} = 1$ — **Zeroth-order signal**

$x^* = t + 3 + \text{noise}$, $\sigma_{noise} = 5$ — **First-order signal**

FORTRAN Program For Conducting Experiments With Zeroth-Order Least Squares Filter

```
      GLOBAL DEFINE
            INCLUDE 'quickdraw.inc'
      END
      IMPLICIT REAL*8 (A-H)
      IMPLICIT REAL*8 (O-Z)
      REAL*8 A(1,1),AINV(1,1),B(1,1),ANS(1,1),X(101),X1(101)
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      SIGNOISE=1.
      N=0
      TS=.1
      SUM3=0.
      SUMPZ1=0.
      SUMPZ2=0.
      DO 10 T=0.,10.,TS
            N=N+1
            CALL GAUSS(XNOISE,SIGNOISE)
            X1(N)=1
            X(N)=X1(N)+XNOISE
            SUM3=SUM3+X(N)
            NMAX=N
10    CONTINUE
      A(1,1)=N
      B(1,1)=SUM3
      AINV(1,1)=1./A(1,1)
      ANS(1,1)=AINV(1,1)*B(1,1)
      DO 11 I=1,NMAX
            T=.1*(I-1)
            XHAT=ANS(1,1)
            ERRX=X1(I)-XHAT
            ERRXP=X(I)-XHAT
            ERRX2=(X1(I)-XHAT)**2
            ERRXP2=(X(I)-XHAT)**2
            SUMPZ1=ERRX2+SUMPZ1
            SUMPZ2=ERRXP2+SUMPZ2
            WRITE(9,*)T,X1(I),X(I),XHAT,ERRX,ERRXP,SUMPZ1,SUMPZ2
            WRITE(1,*)T,X1(I),X(I),XHAT,ERRX,ERRXP,SUMPZ1,SUMPZ2
11    CONTINUE
      CLOSE(1)
      PAUSE
      END
```

Measurement noise standard deviation (SIGNOISE=1.)

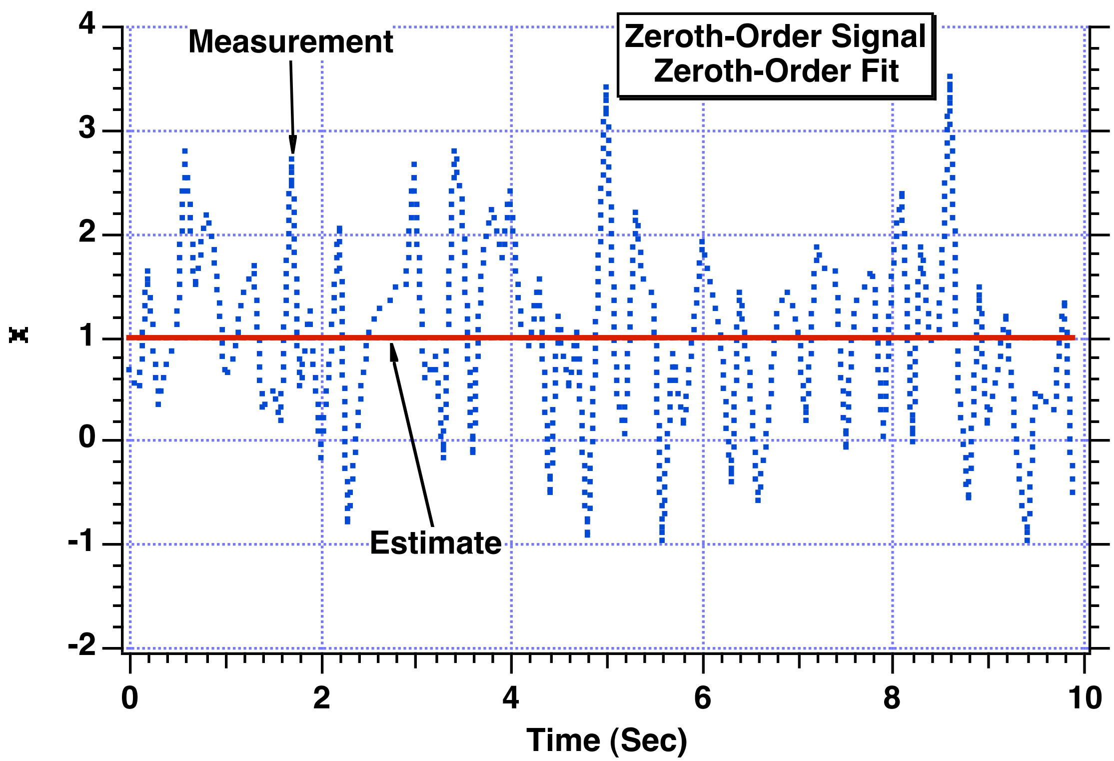
Actual signal (X1(N)=1)

Measurement (X(N)=X1(N)+XNOISE)

Zeroth-order filter (A(1,1)=N through ANS(1,1)=AINV(1,1)*B(1,1))

Error computation (XHAT=ANS(1,1) through SUMPZ2=ERRXP2+SUMPZ2)

Zeroth-Order Filter Smoothes Noisy Measurements

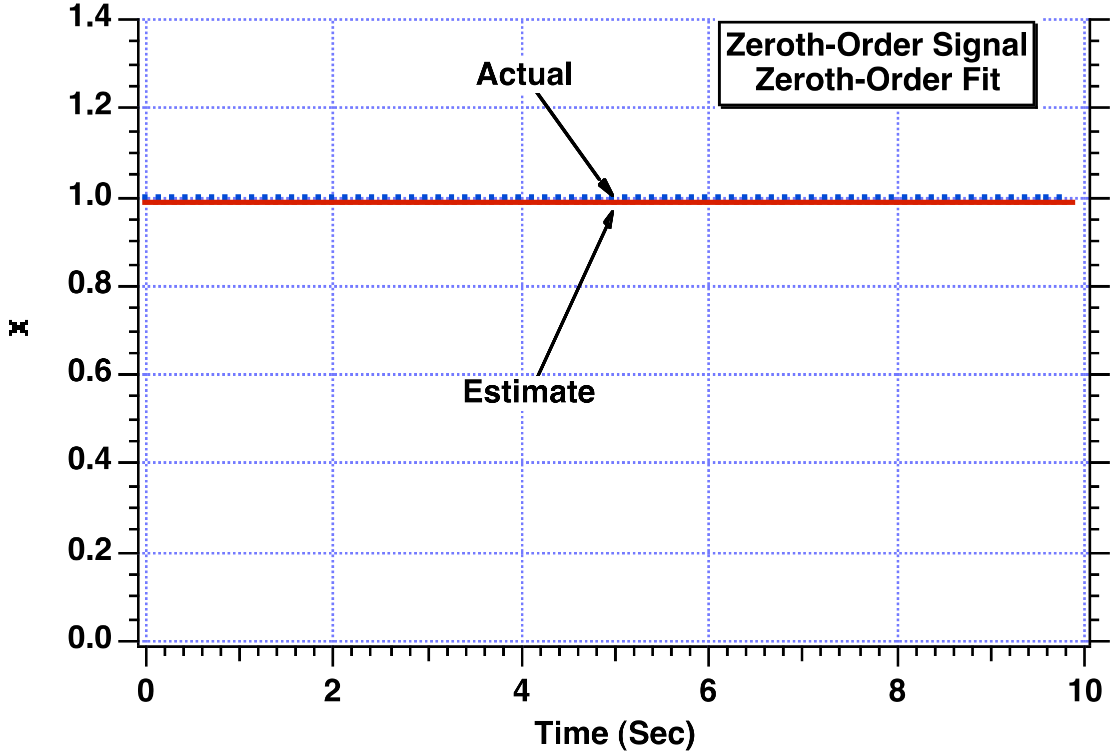


Measurement

$x^* = 1 + \text{noise}$

$\sigma_{noise} = 1$

Zeroth-Order Filter Yields Near Perfect Estimates of Constant Signal

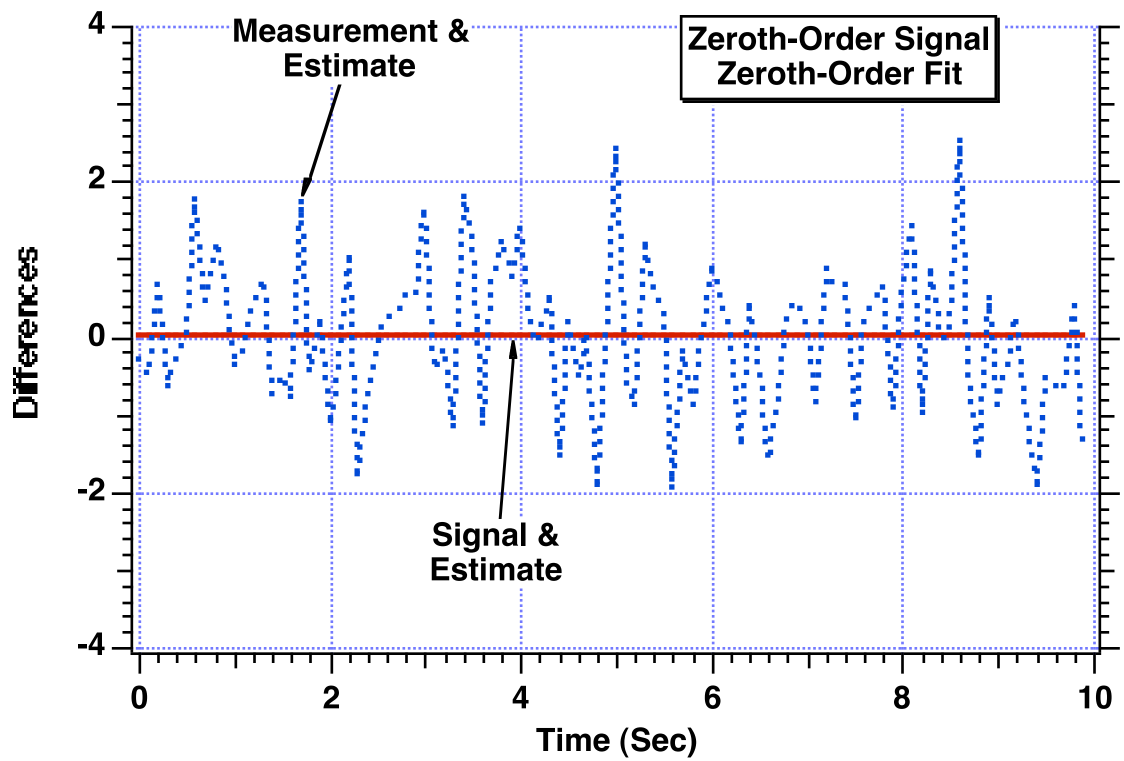


Measurement

$x^* = 1 + \text{noise}$

$\sigma_{noise} = 1$

Estimation Errors Are Nearly Zero For Zeroth-Order Least Squares Filter



$$\sum (\text{Signal - Estimate})^2 = .01507$$

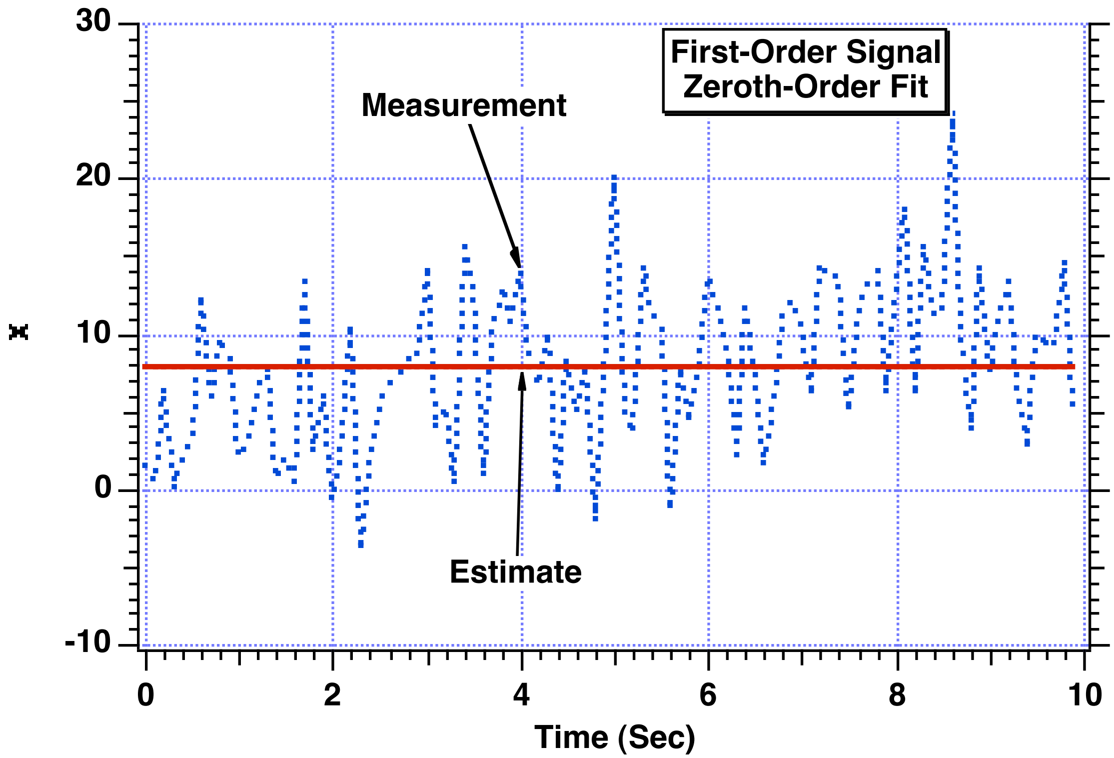
$$\sum (\text{Measurement - Estimate})^2 = 91.92$$

Increasing Order of Signal and Changing Noise Standard Deviation

Measurement

$$x^* = t + 3 + noise$$
$$\sigma_{noise} = 5$$

Zeroth-Order Least Squares Filter Does Not Capture Upward Trend of Measurement Data

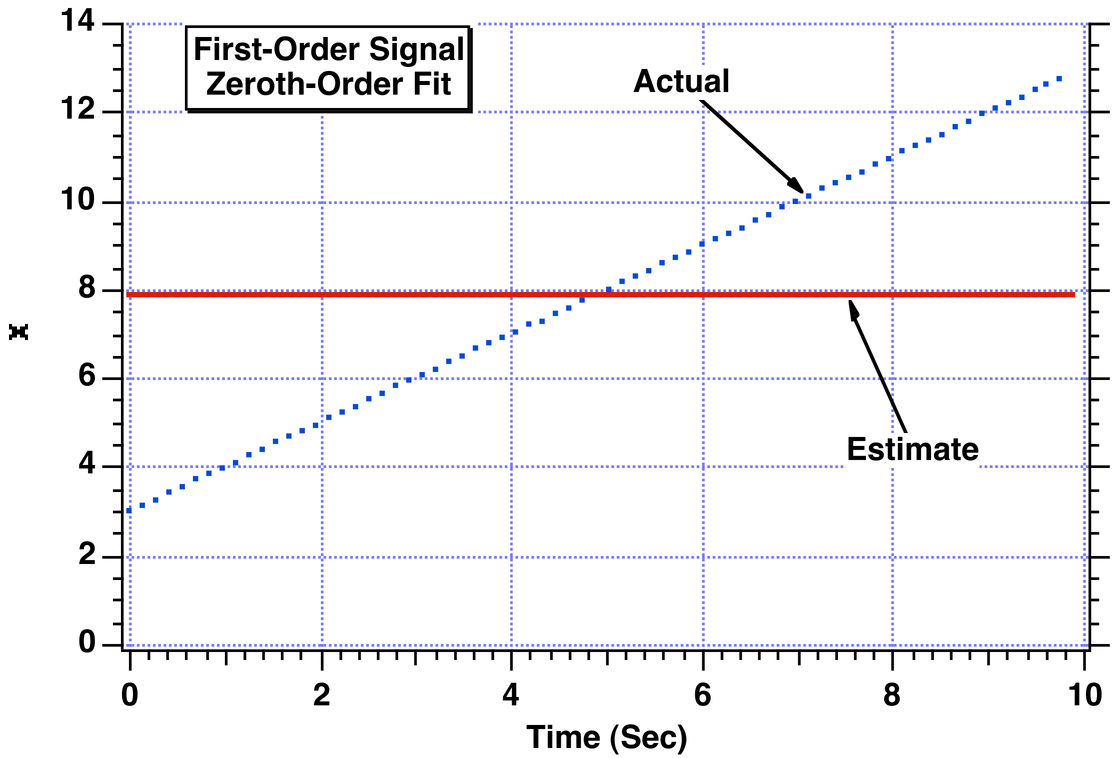


Measurement

$$x^* = t + 3 + \text{noise}$$

$$\sigma_{noise} = 5$$

Zeroth-Order Least Squares Filter Can Not Estimate Slope of Signal

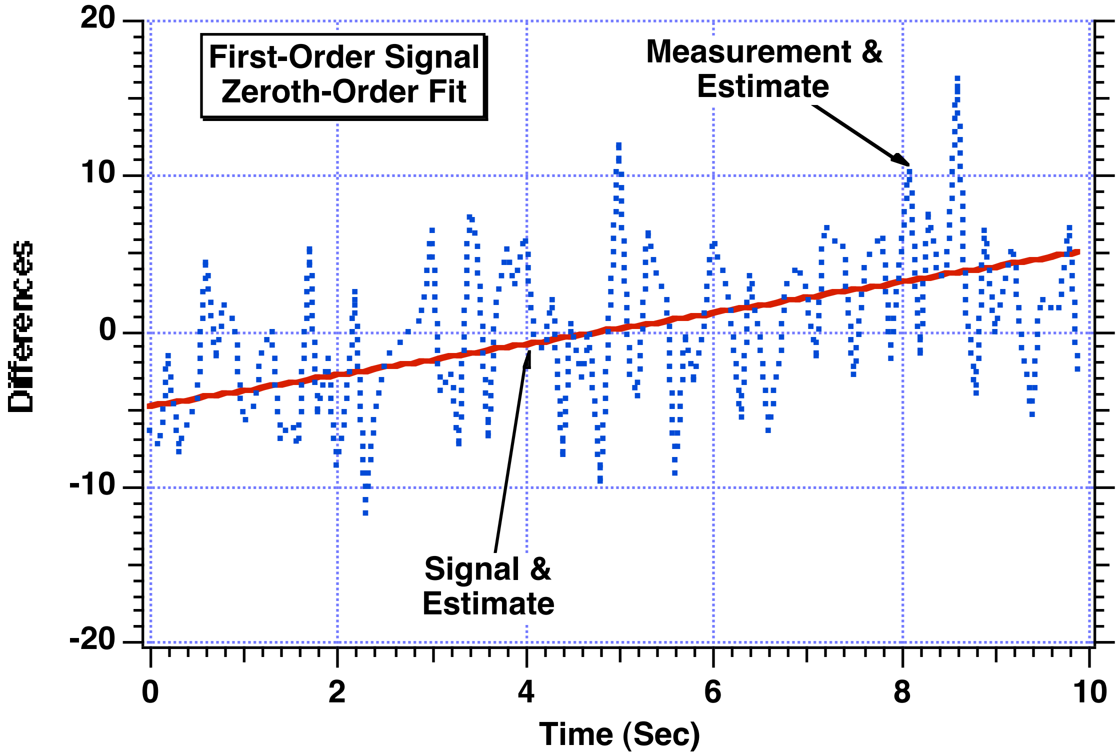


Measurement

$$x^* = t + 3 + \text{noise}$$

$$\sigma_{noise} = 5$$

Errors in Estimate of Signal Grow With Time



$\sum$ (Signal - Estimate)^2 = 834

$\sum$ (Measurement - Estimate)^2 = 2736

→ **Larger values indicate filter is diverging**

Experiments With First-Order or Two-State Least Squares Filter

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} (k-1)T_s \\ \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} x_k^* \\ \sum_{k=1}^{n} (k-1)T_s x_k^* \end{bmatrix}$$

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

$$\hat{\dot{x}}_k = a_1$$

Measurements considered

$x^* = 1 + \text{noise}$, $\sigma_{noise} = 1$ — **Zeroth-order signal**

$x^* = t + 3 + \text{noise}$, $\sigma_{noise} = 5$ — **First-order signal**

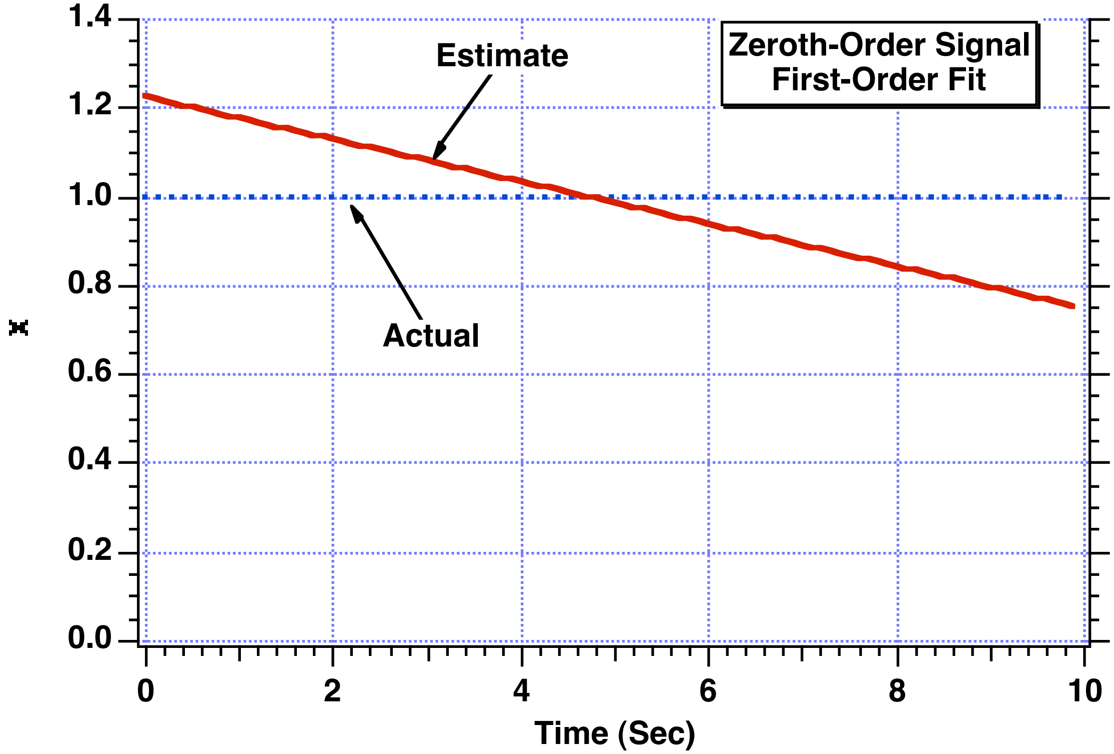
$x^* = 5t^2 - 2t + 2 + \text{noise}$, $\sigma_{noise} = 50$ — **Second-order signal**

MATLAB Code For Conducting Experiments With First-Order Least Squares Filter

Measurement noise standard deviation (SIGNOISE); Actual signal (X1(N)=1); Measurement (X(N)=X1(N)+XNOISE); First-order filter (A, B, AINV, ANS); Errors (ERRX … SUMPZ2)

```
SIGNOISE=1.;
N=0;
TS=.1;
SUM1=0;
SUM2=0;
SUM3=0;
SUM4=0.;
SUMPZ1=0.;
SUMPZ2=0.;
count=0;
for T=0:TS:10
        N=N+1;
        XNOISE=SIGNOISE*randn;
        X1(N)=1;
        XD(N)=0.;
        X(N)=X1(N)+XNOISE;
        SUM1=SUM1+T;
        SUM2=SUM2+T*T;
        SUM3=SUM3+X(N);
        SUM4=SUM4+T*X(N);
        NMAX=N;
end
A(1,1)=N;
A(1,2)=SUM1;
A(2,1)=SUM1;
A(2,2)=SUM2;
B(1,1)=SUM3;
B(2,1)=SUM4;
AINV=inv(A);
ANS=AINV*B;
for I=1:NMAX
        T=.1*(I-1);
        XHAT=ANS(1,1)+ANS(2,1)*T;
        XDHAT=ANS(2,1);
        ERRX=X1(I)-XHAT;
        ERRXD=XD(I)-XDHAT;
        ERRXP=X(I)-XHAT;
        ERRX2=(X1(I)-XHAT)^2;
        ERRXP2=(X(I)-XHAT)^2;
        SUMPZ1=ERRX2+SUMPZ1;
        SUMPZ2=ERRXP2+SUMPZ2;
        count=count+1;
        ArrayT(count)=T;
        ArrayA(count)=X1(I);
        ArrayB(count)=X(I);
        ArrayXHAT(count)=XHAT;
        ArrayERRX(count)=ERRX;
        ArrayERRXD(count)=ERRXD;
        ArraySUMPZ1(count)=SUMPZ1;
        ArraySUMPZ2(count)=SUMPZ2;
end
clc
output=[ArrayT',ArrayA',ArrayB',ArrayXHAT',ArrayERRX',ArrayERRXD',ArraySUMPZ1',ArraySUMPZ2'];
save datfil output -ascii
disp 'simulation finished'
```

First-Order Filter Has Trouble in Estimating Zeroth-Order Signal

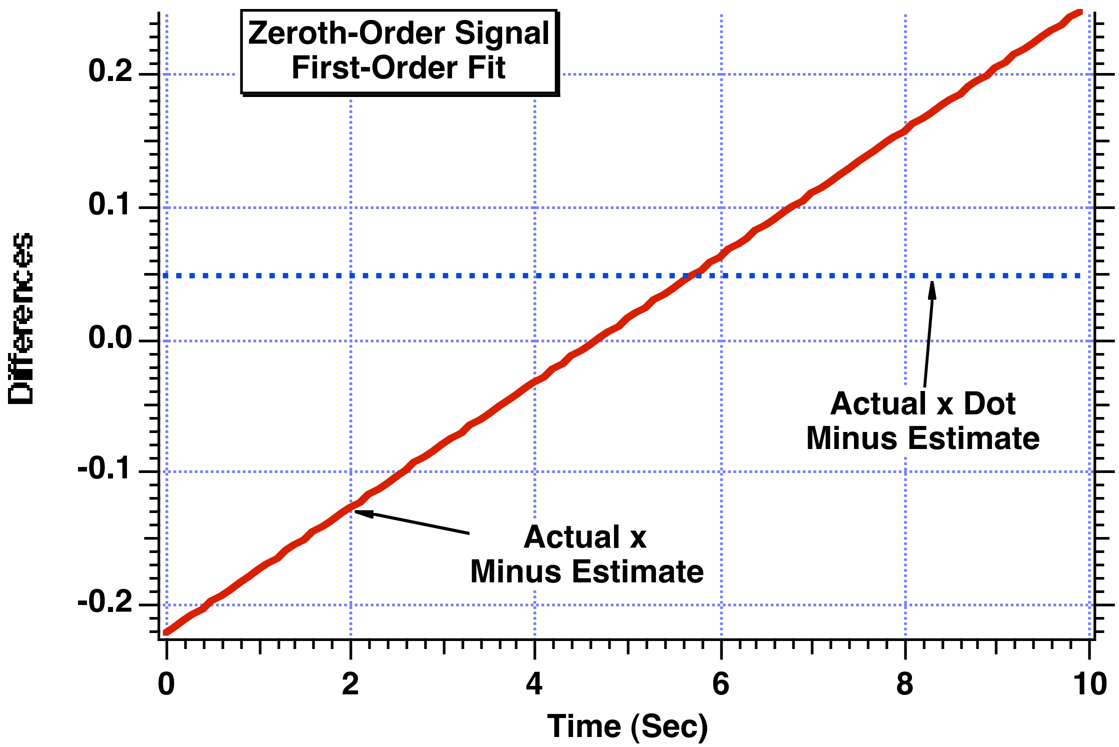


Measurement

$x^* = 1 + \text{noise}$

$\sigma_{noise} = 1$

Errors in Estimate of Signal and It's Derivative Are Not Too Large



$$\sum (\text{Signal - Estimate})^2 = 1.895$$

$$\sum (\text{Measurement - Estimate})^2 = 90.04$$

Performing worse than zeroth-order filter

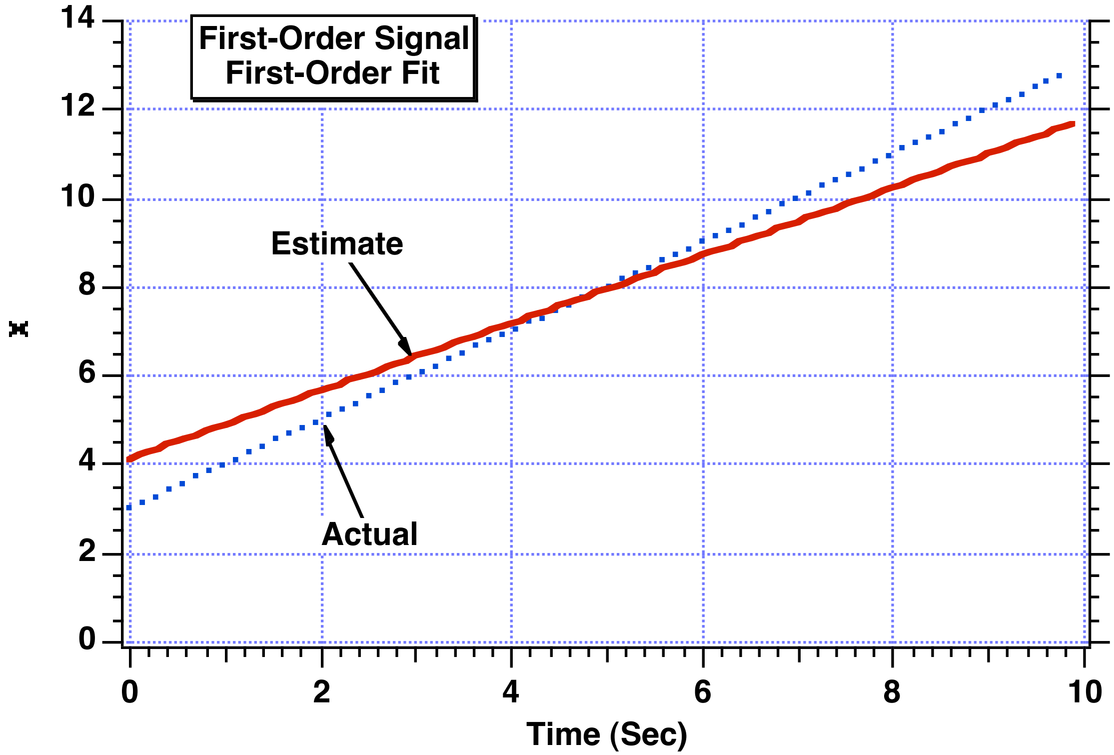
Increasing Order of Signal and Changing Noise Standard Deviation

Measurement

$$x^* = t + 3 + noise$$

$$\sigma_{noise} = 5$$

First-Order Filter Does Much Better Job in Estimating First-Order Signal Than Zeroth-Order Filter

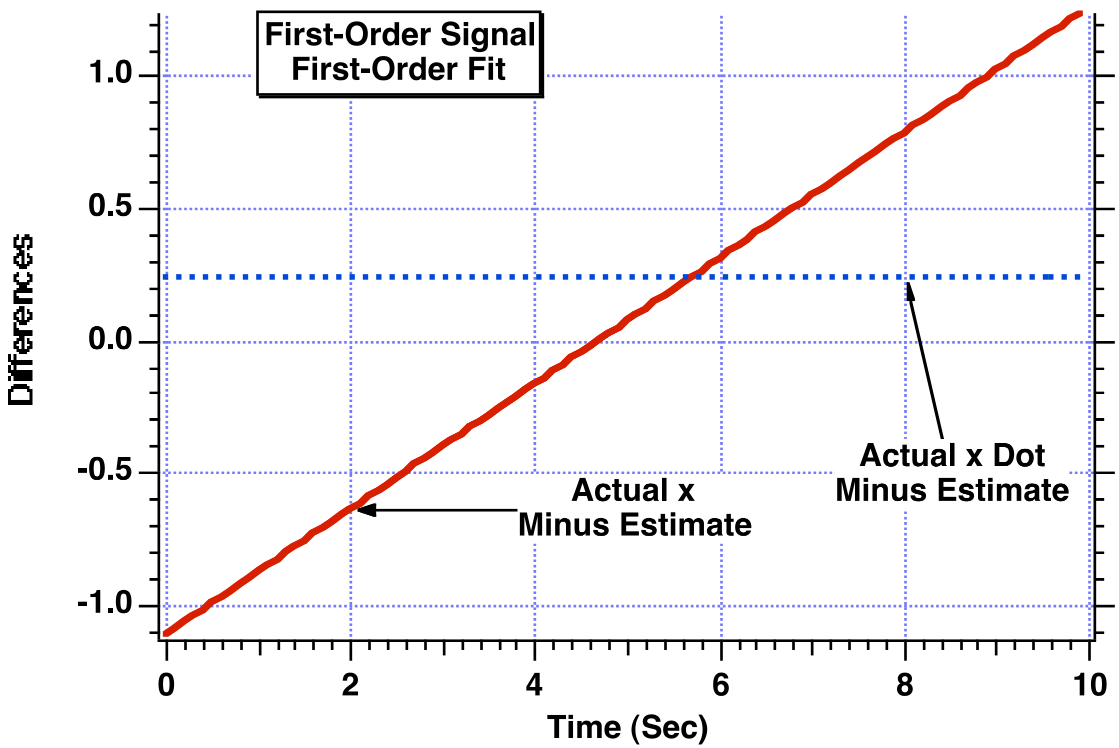


Measurement

$$x^* = t + 3 + noise$$

$$\sigma_{noise} = 5$$

First-Order Filter is Able To Estimate Derivative of First-Order Signal Accurately



$\sum$ (Signal - Estimate)^2 = 47.38 ← **Much better than zeroth-order filter**

$\sum$ (Measurement - Estimate)^2 = 2251

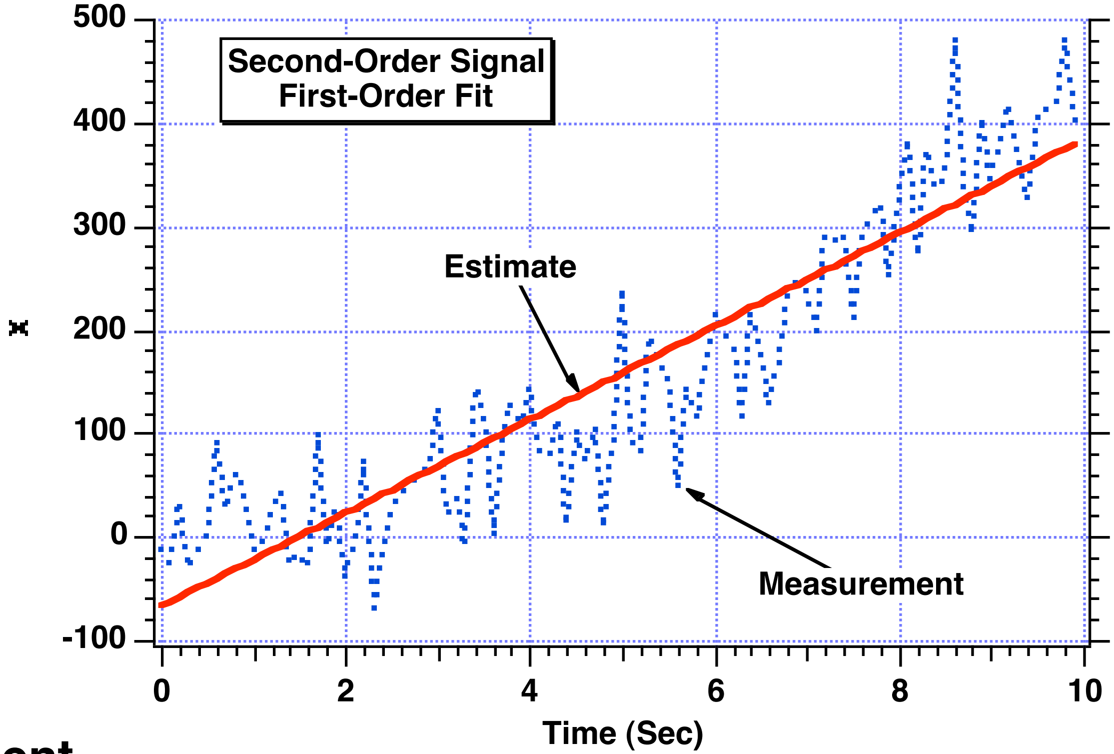
Increasing Order of Signal and Changing Noise Standard Deviation

Measurement

$x^* = 5t^2 - 2t + 2 + \text{noise}$

$\sigma_{noise} = 50$

First-Order Filter Attempts to Track Second-Order Measurements

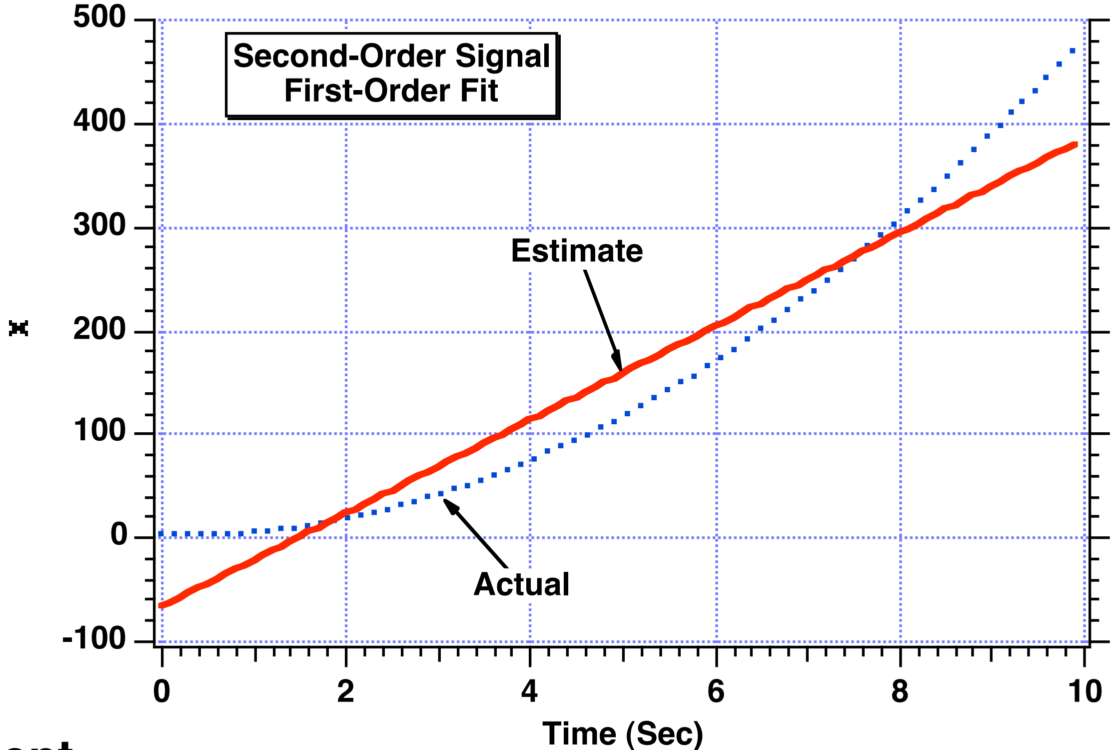


Measurement

$$x^* = 5t^2 - 2t + 2 + \text{noise}$$

$$\sigma_{noise} = 50$$

On the Average First-Order Filter Estimates Second-Order Signal

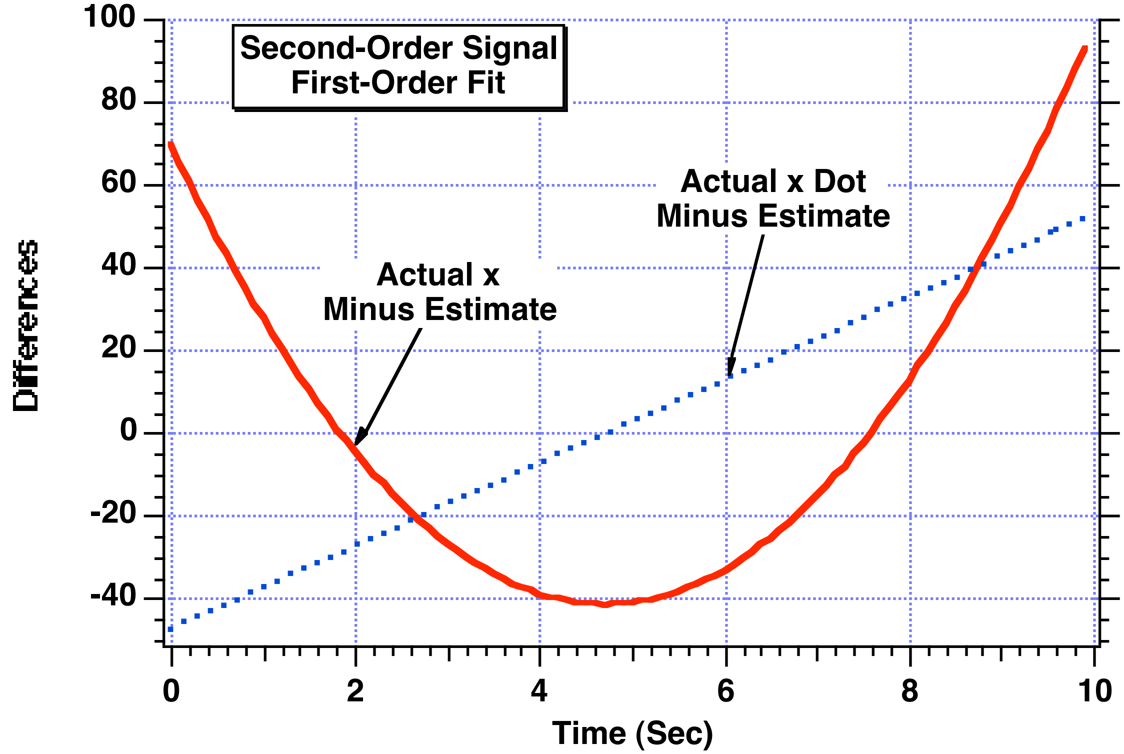


Measurement

$$x^* = 5t^2 - 2t + 2 + \text{noise}$$

$$\sigma_{noise} = 50$$

Large Estimation Errors Result When First-Order Filter Attempts to Track Second-Order Signal



$$\sum (\text{Signal - Estimate})^2 = 143557$$

$$\sum (\text{Measurement - Estimate})^2 = 331960$$

→ **Larger Values Indicate Filter Is Diverging**

Experiments With Second-Order or Three-State Least Squares Filter

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 \\ \sum_{k=1}^{n} (k-1)T_s & \sum_{k=1}^{n} [(k-1)T_s]^2 & \sum_{k=1}^{n} [(k-1)T_s]^3 \\ \sum_{k=1}^{n} [(k-1)T_s]^2 & \sum_{k=1}^{n} [(k-1)T_s]^3 & \sum_{k=1}^{n} [(k-1)T_s]^4 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} x_k^* \\ \sum_{k=1}^{n} (k-1)T_s x_k^* \\ \sum_{k=1}^{n} [(k-1)T_s]^2 x_k^* \end{bmatrix}$$

$$\hat{x}_k = a_0 + a_1(k-1)T_s + a_2[(k-1)T_s]^2$$

$$\hat{\dot{x}}_k = a_1 + 2a_2(k-1)T_s$$

$$\hat{\ddot{x}}_k = 2a_2$$

Measurements considered

$x^* = 1 + \text{noise}$, $\sigma_{noise} = 1$ — **Zeroth-order signal**

$x^* = t + 3 + \text{noise}$, $\sigma_{noise} = 5$ — **First-order signal**

$x^* = 5t^2 - 2t + 2 + \text{noise}$, $\sigma_{noise} = 50$ — **Second-order signal**

MATLAB Code For Conducting Experiments With Second-Order Least Squares Filter - 1

```
SIGNOISE=1.;
TS=.1;
N=0;
SUM1=0.;
SUM2=0.;
SUM3=0.;
SUM4=0.;
SUM5=0.;
SUM6=0.;
SUM7=0.;
SUMPZ1=0.;
SUMPZ2=0.;
count=0;
for T=0:TS:10
	N=N+1;
	XNOISE=SIGNOISE*randn;
	X1(N)=1.;
	XD(N)=0.;
	XDD(N)=0.;
	X(N)=X1(N)+XNOISE;
	SUM1=SUM1+T;
	SUM2=SUM2+T*T;
	SUM3=SUM3+X(N);
	SUM4=SUM4+T*X(N);
	SUM5=SUM5+T^3;
	SUM6=SUM6+T^4;
	SUM7=SUM7+T*T*X(N);
	NMAX=N;
end
A(1,1)=N;
A(1,2)=SUM1;
A(1,3)=SUM2;
A(2,1)=SUM1;
A(2,2)=SUM2;
A(2,3)=SUM5;
A(3,1)=SUM2;
A(3,2)=SUM5;
A(3,3)=SUM6;
B(1,1)=SUM3;
B(2,1)=SUM4;
B(3,1)=SUM7;
AINV=inv(A);
ANS=AINV*B;
```

Measurement noise standard deviation (SIGNOISE=1.;)

Signal (X1(N)=1.;)

Measurement (X(N)=X1(N)+XNOISE;)

Solving for second-order filter coefficients (A(1,1)=N; … ANS=AINV*B;)

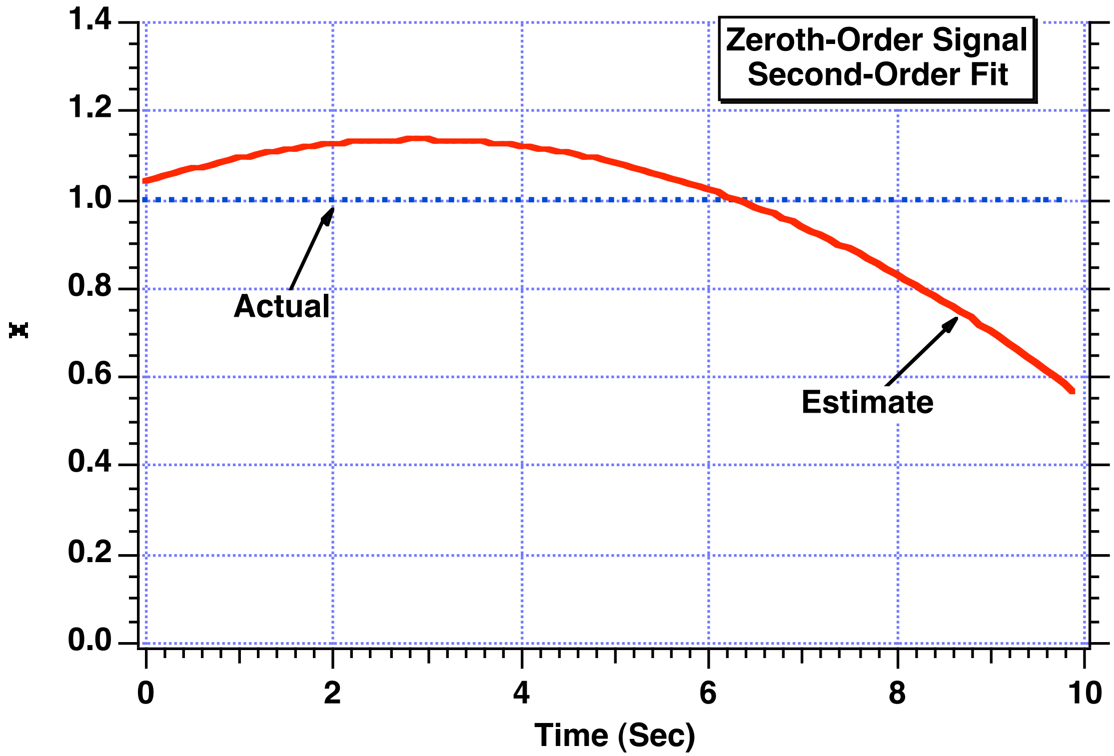
MATLAB Code For Conducting Experiments With Second-Order Least Squares Filter - 2

```
for I=1:NMAX
            T=.1*(I-1);
            XHAT=ANS(1,1)+ANS(2,1)*T+ANS(3,1)*T*T;
            XDHAT=ANS(2,1)+2.*ANS(3,1)*T;
            XDDHAT=2.*ANS(3,1);
            ERRX=X1(I)-XHAT;
            ERRXD=XD(I)-XDHAT;
            ERRXDD=XDD(I)-XDDHAT;
            ERRXP=X(I)-XHAT;
            ERRX2=(X1(I)-XHAT)^2;
            ERRXP2=(X(I)-XHAT)^2;
            SUMPZ1=ERRX2+SUMPZ1;
            SUMPZ2=ERRXP2+SUMPZ2;
            count=count+1;
            ArrayT(count)=T;
            ArrayA(count)=X1(I);
            ArrayB(count)=X(I);
            ArrayXHAT(count)=XHAT;
            ArrayERRX(count)=ERRX;
            ArrayERRXD(count)=ERRXD;
            ArrayERRXDD(count)=ERRXDD;
            ArraySUMPZ1(count)=SUMPZ1;
            ArraySUMPZ2(count)=SUMPZ2;
end
figure
plot(ArrayT,ArrayA,ArrayT,ArrayXHAT),grid
xlabel('Time (Sec)')
ylabel('Estimates and Actual')
axis([0 10 0 1.4])
figure
plot(ArrayT,ArrayERRX,ArrayT,ArrayERRXD,ArrayT,ArrayERRXDD),grid
xlabel('Time (Sec)')
ylabel('Differences')
axis([0 10 -.2 .5])
clc
output=[ArrayT',ArrayA',ArrayB',ArrayXHAT',ArrayERRX',ArrayERRXD',ArrayERRXDD',ArraySUMPZ1',ArraySUMPZ2'];
save datfil output  -ascii
disp 'simulation finished'
```

State estimates (lines T= through XDHAT=)

Errors (lines ERRX= through SUMPZ2=)

Second-Order Filter Estimates Signal is Parabola Even Though it is a Constant

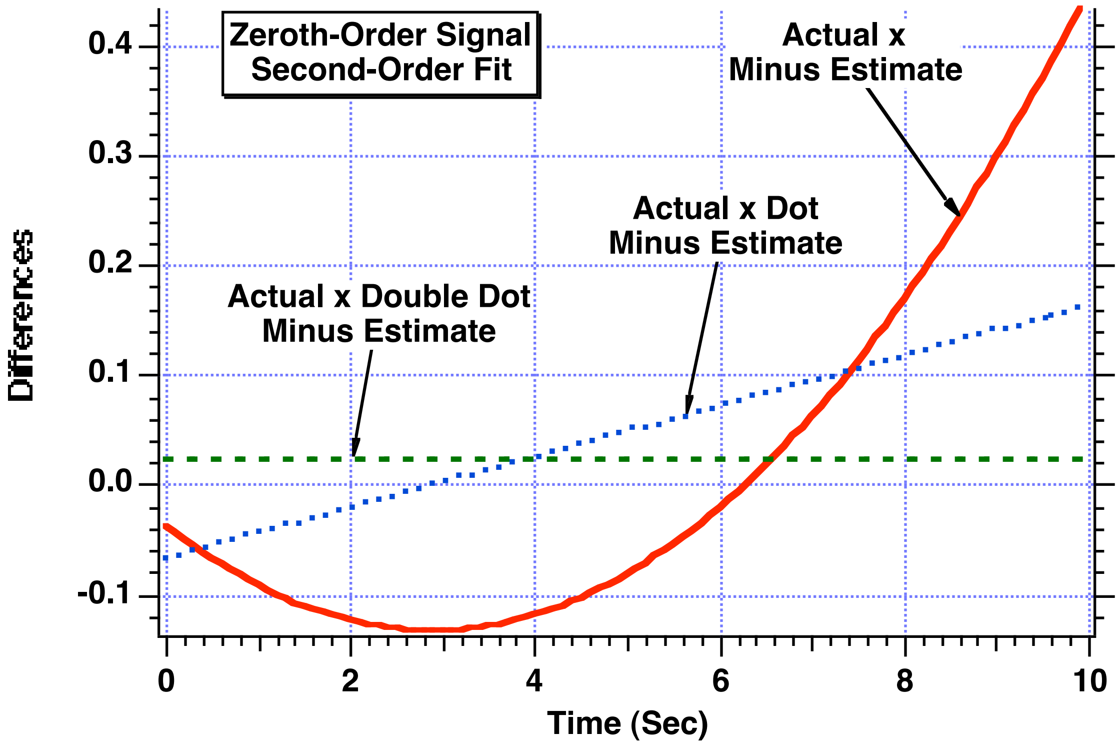


Measurement

$x^* = 1 + \text{noise}$

$\sigma_{noise} = 1$

Estimation Errors Between Estimates and States of Signal Are Not Terrible When Order of Filter is Too High



$\sum$ (Signal - Estimate)^2 = 2.63 ← **Larger than zeroth and first-order filters**

$\sum$ (Measurement - Estimate)^2 = 89.3 ← **Smaller than zeroth and first-order filters**

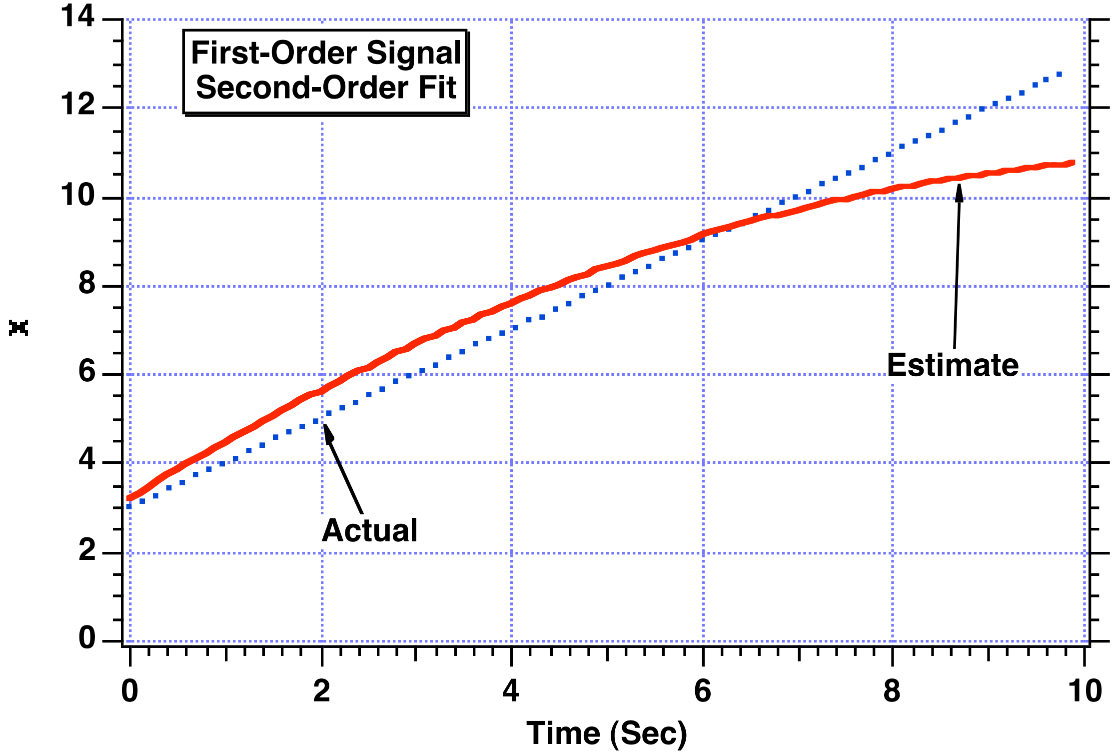
Increasing Order of Signal and Changing Noise Standard Deviation

Measurement

$x^* = t + 3 + noise$

$\sigma_{noise} = 1$

Second-Order Filter Attempts to Fit First-Order Signal With a Parabola

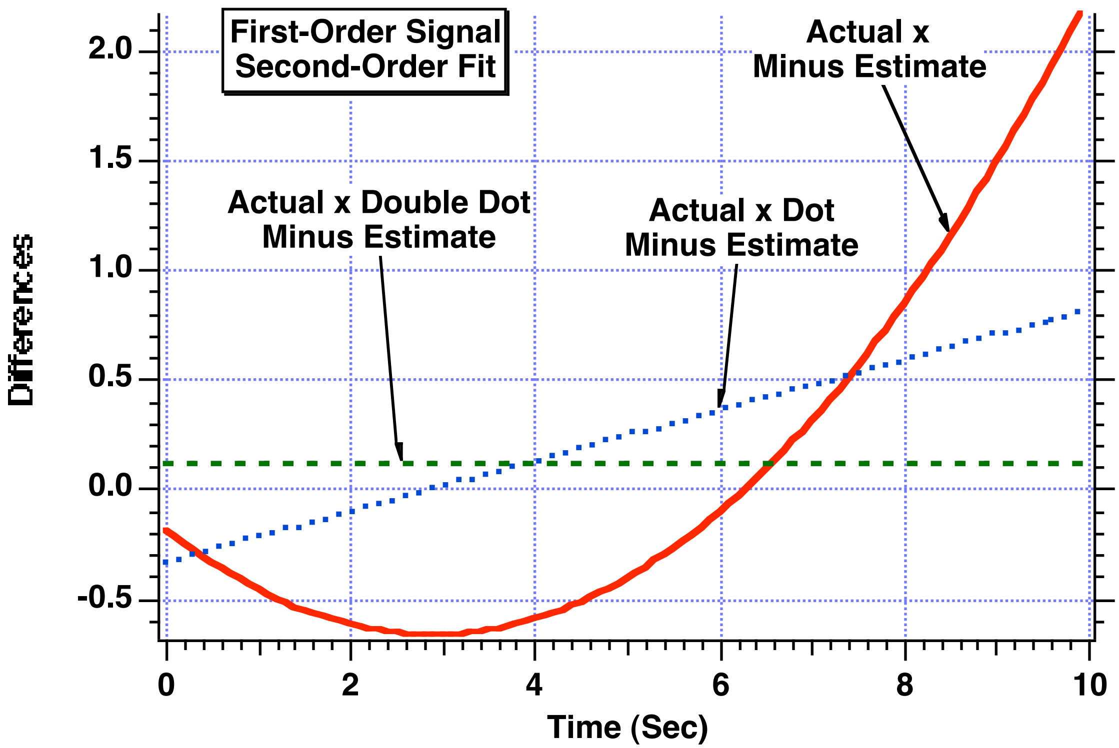


Measurement

$$x^* = t + 3 + noise$$

$$\sigma_{noise} = 1$$

Second-Order Fit to First-Order Signal Yields Larger Errors Than First-Order Fit



$\sum$ (Signal - Estimate)^2 = 65.8 ← **Larger than first-order filter**

$\sum$ (Measurement - Estimate)^2 = 2232 ← **Smaller than first-order filter**

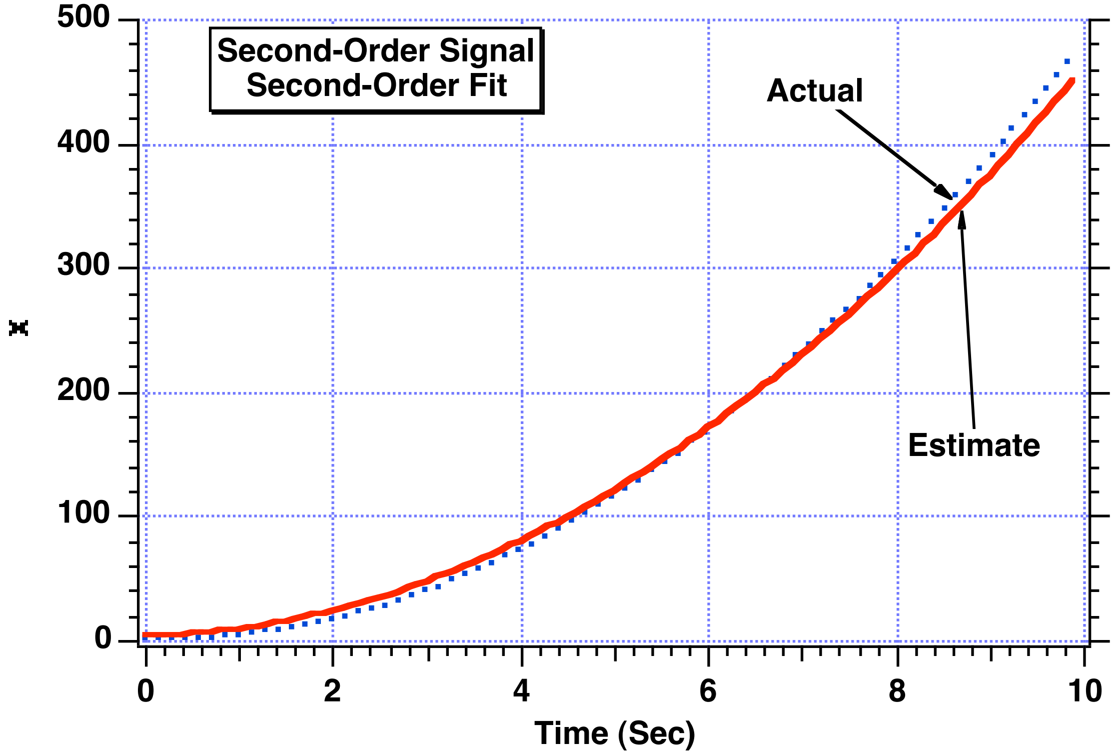
Increasing Order of Signal and Changing Noise Standard Deviation

Measurement

$x^* = 5t^2 - 2t + 2 + \text{noise}$

$\sigma_{noise} = 50$

Second-Order Filter Provides Near Perfect Estimates of Second-Order Signal

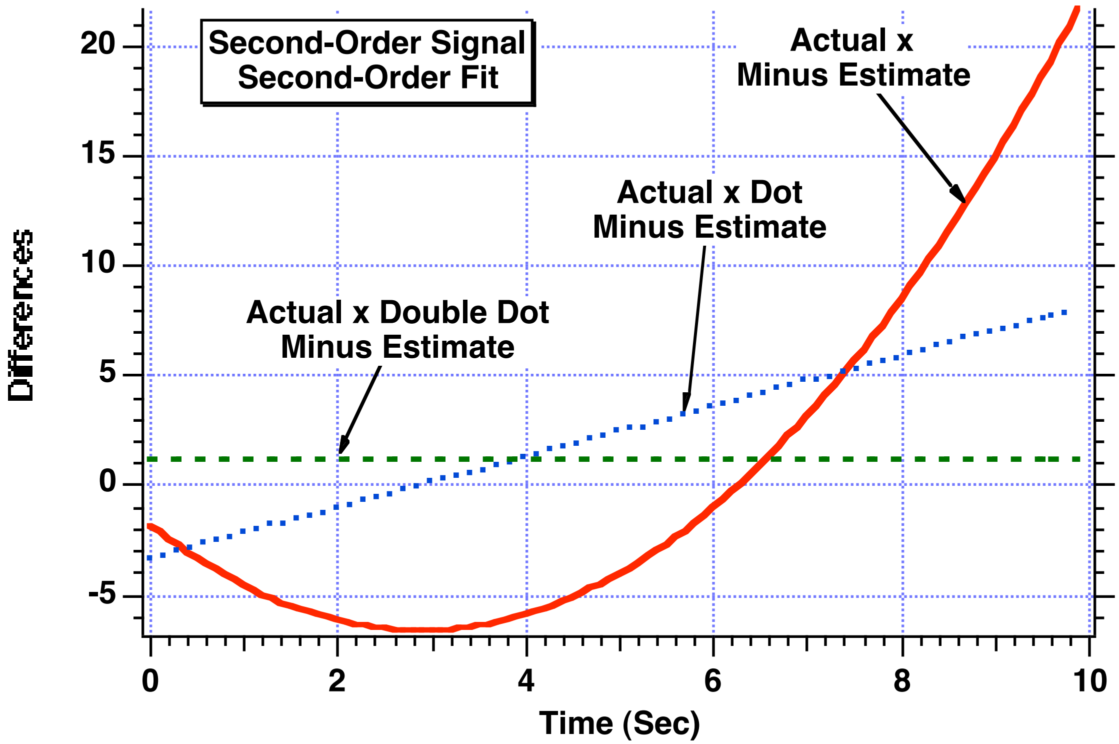


Measurement

$x^* = 5t^2 - 2t + 2 + \text{noise}$

$\sigma_{noise} = 50$

The Error in the Estimates of All States of Second-Order Filter Against Second-Order Signal are Better Than All Other Filter Fits



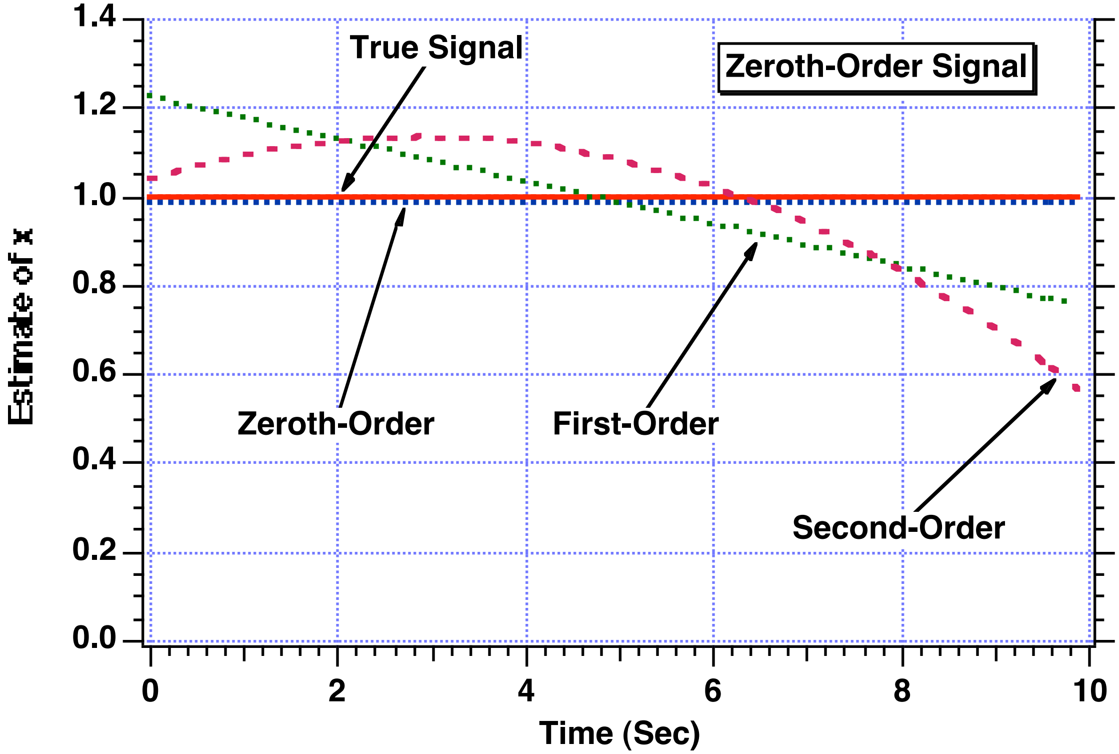
$\sum$ (Signal - Estimate)^2 = 6577.

Both smaller than first-order filter

$\sum$ (Measurement - Estimate)^2 = 223265

Comparison of Filters

Zeroth-Order Least Squares Filter Best Tracks Zeroth-Order Measurement

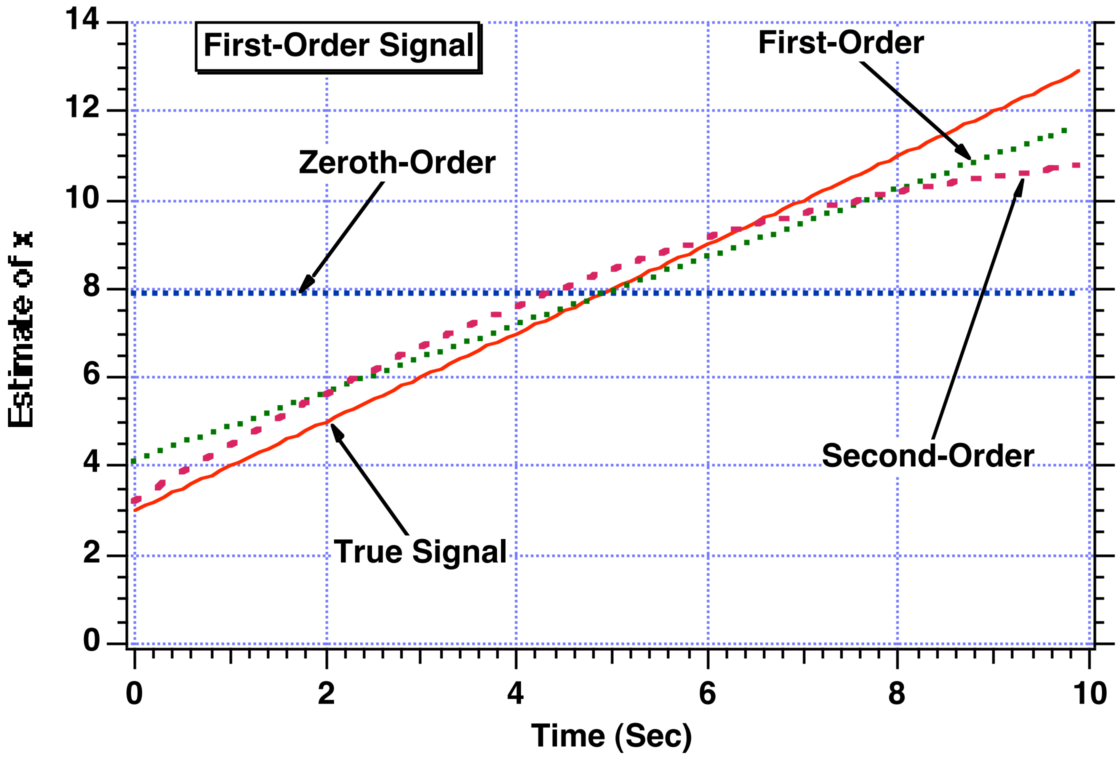


Measurement

$x^* = 1 + \text{noise}$

$\sigma_{noise} = 1$

First-Order Least Squares Filter Best Tracks First-Order Measurement

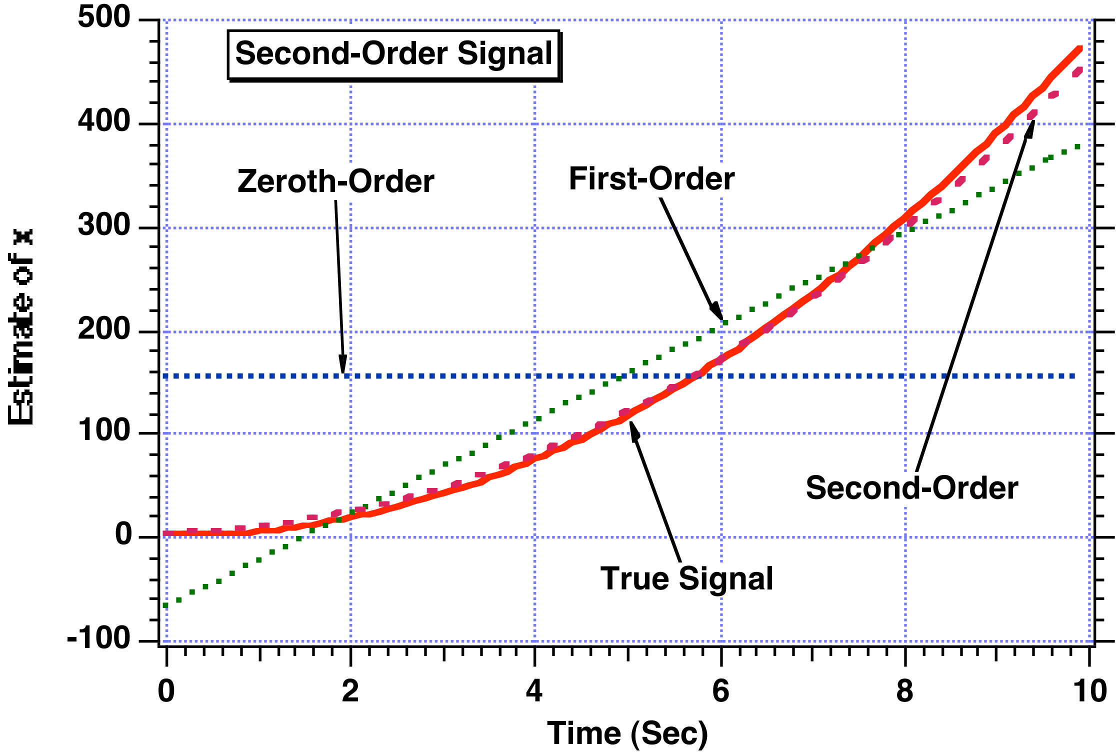


Measurement

$x^* = t + 3 + \text{noise}$

$\sigma_{noise} = 1$

Second-Order Least Squares Filter Tracks Parabolic Signal Quite Well



Measurement

$x^* = 5t^2 - 2t + 2 + \text{noise}$

$\sigma_{noise} = 50$

From a Quantitative Point of View Best Estimates of Signal are Obtained When Filter Order Matches Signal Order

$\sum$ (Signal - Estimate)^2

Filter Order \ Signal Order	0	1	2
0	.01057	834	
1	1.895	47.38	143557
2	2.63	65.8	6577

***Note that diagonal elements are smallest**

From a Quantitative Point of View Estimates Get Closer To Measurements When Filter Order Gets Higher

$\sum$ (Measurement - Estimate)^2

Filter Order \ Signal Order	0	1	2
0	91.92	2736	
1	90.04	2251	331960
2	89.3	2232	223265

*** Note that last row is smallest**

Accelerometer Testing Example

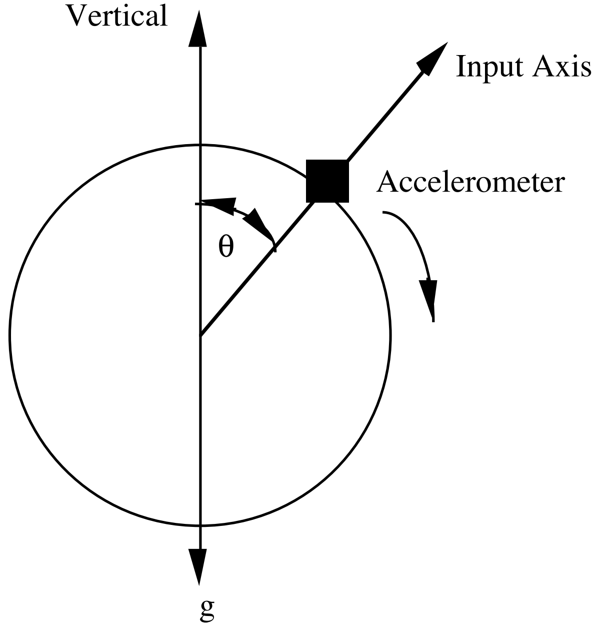
General Least Squares Coefficients For Different Order Polynomial Fits

Before $\hat{x}_k = a_0 + a_1(k-1)T_s + a_2[(k-1)T_s]^2 + ... + a_n[(k-1)T_s]^n$

In general $\hat{y} = a_0 + a_1 x + a_2 x^2 + ... + a_n x^n$

Order	Equations
Zeroth	$a_0 = \dfrac{\sum_{k=1}^{n} y_k^*}{n}$
First	$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} x_k \\ \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} y_k^* \\ \sum_{k=1}^{n} x_k y_k^* \end{bmatrix}$
Second	$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 \\ \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k^3 \\ \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k^3 & \sum_{k=1}^{n} x_k^4 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} y_k^* \\ \sum_{k=1}^{n} x_k y_k^* \\ \sum_{k=1}^{n} x_k^2 y_k^* \end{bmatrix}$

Accelerometer Experiment Test Setup



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

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

Formulating Error Equations For Least Squares Filter

Error equation for perfect angular measurements

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

Error equation for noisy angular measurements

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

For filter implementation

$$\text{Error} \rightarrow y_k$$

$$g\cos\theta_k^* \rightarrow x_k$$

It is important to note that

$$g\cos\theta_K^* - g\cos\theta_K \neq 0$$

Nominal Values For Accelerometer Testing Example

Term	Scientific Value	English Units
Bias Error	10 μg	$10*10^{-6}*32.2=.000322$ ft/sec^2
Scale Factor Error	5 ppm	$5*10^{-6}$
G-Squared Sensitive Drift	1 μg/g^2	$1*10^{-6}/32.2=3.106*10^{-8}$ sec^2/ft

Filter Formulation

Measurement

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

Independent variable

$$x_k = g\cos\theta_k^*$$

Use second-order fit to data because measurement appears to be second-order

$$\hat{y}_k = a_0 + a_1 x_k + a_2 x_k^2$$

Filter formula

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} n & \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 \\ \sum_{k=1}^{n} x_k & \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k^3 \\ \sum_{k=1}^{n} x_k^2 & \sum_{k=1}^{n} x_k^3 & \sum_{k=1}^{n} x_k^4 \end{bmatrix}^{-1} \begin{bmatrix} \sum_{k=1}^{n} y_k^* \\ \sum_{k=1}^{n} x_k y_k^* \\ \sum_{k=1}^{n} x_k^2 y_k^* \end{bmatrix} \longrightarrow \begin{array}{l} \hat{B} = a_0 \\ \widehat{SF} = a_1 \\ \hat{K} = a_2 \end{array}$$

Method of Least Squares Applied to Accelerometer Testing Problem - 1

```
BIAS=.00001*32.2;
SF=.000005;
XK=.000001/32.2;
SIGTH=0.;
G=32.2;
JJ=0;
count=0;
for THETDEG=0:2:180
        THET=THETDEG/57.3;
        THETNOISE=SIGTH*randn;
        THETS=THET+THETNOISE;
        JJ=JJ+1;
        T(JJ)=32.2*cos(THETS);
        X(JJ)=BIAS+SF*G*cos(THETS)+XK*(G*cos(THETS))^2-G*cos(THET)+G*cos(THETS);
end
N=JJ;
SUM1=0;
SUM2=0;
SUM3=0;
SUM4=0;
SUM5=0;
SUM6=0;
SUM7=0;
for I=1:JJ
        SUM1=SUM1+T(I);
        SUM2=SUM2+T(I)*T(I);
        SUM3=SUM3+X(I);
        SUM4=SUM4+T(I)*X(I);
        SUM5=SUM5+T(I)*T(I)*T(I);
        SUM6=SUM6+T(I)*T(I)*T(I)*T(I);
        SUM7=SUM7+T(I)*T(I)*X(I);
end
```

Generating measurement data

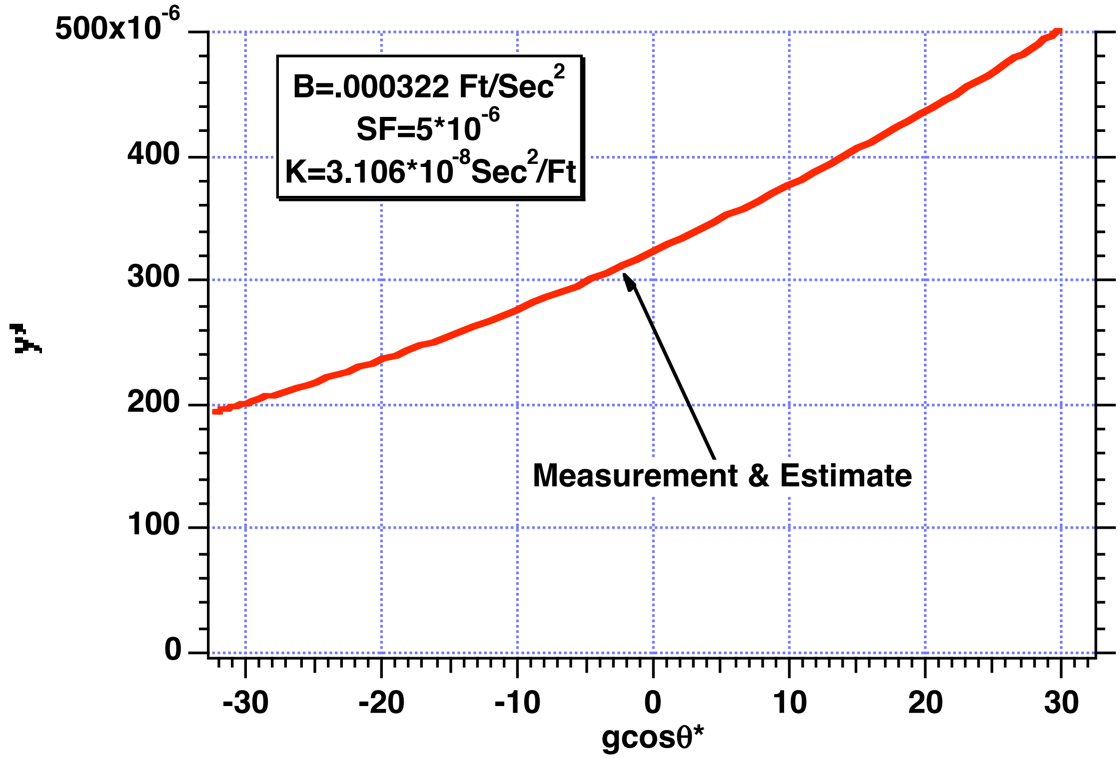
Method of Least Squares Applied to Accelerometer Testing Problem - 2

Second-order least squares filter (applies to the lines from A(1,1)=N; through ANS=AINV*B)

Filter estimate (points to the PZ(JJ)= line)

```
A(1,1)=N;
A(1,2)=SUM1;
A(1,3)=SUM2;
A(2,1)=SUM1;
A(2,2)=SUM2;
A(2,3)=SUM5;
A(3,1)=SUM2;
A(3,2)=SUM5;
A(3,3)=SUM6;
AINV=inv(A);
B(1,1)=SUM3;
B(2,1)=SUM4;
B(3,1)=SUM7;
ANS=AINV*B
for JJ=1:N
        PZ(JJ)=ANS(1,1)+ANS(2,1)*T(JJ)+ANS(3,1)*T(JJ)*T(JJ);
        count=count+1;
        ArrayA(count)=T(JJ);
        ArrayB(count)=X(JJ);
        ArrayPZ(count)=PZ(JJ);
end
figure
plot(ArrayA,ArrayB,ArrayA,ArrayPZ),grid
xlabel('gcos(thet) (deg)')
ylabel('Measurement and Estimate')
axis([-35 35 0 .0005])
clc
output=[ArrayA',ArrayB',ArrayPZ'];
save datfil output  -ascii
disp 'simulation finished'
```

Without Measurement Noise We Can Estimate Accelerometer Errors Perfectly



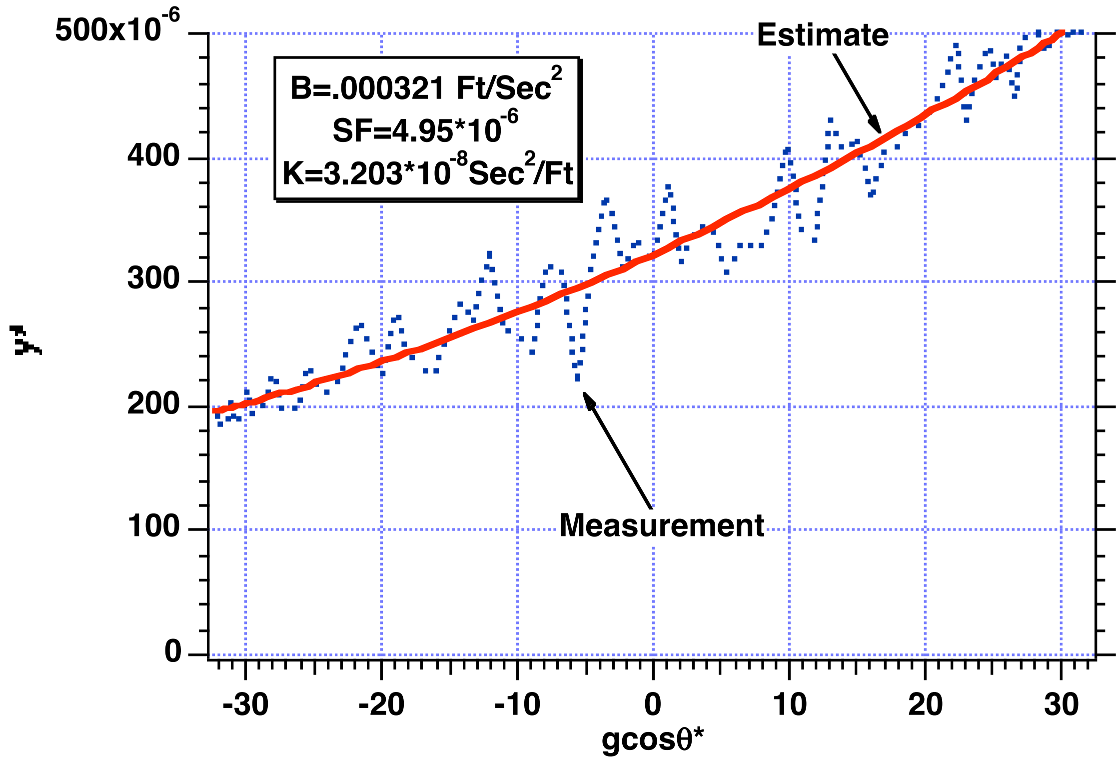
Truth

$B = .000322\ ft/sec^2$

$SF = 5*10^{-6}$

$K = 3.106*10^{-8}\ sec^2/ft$

With 1 μR of Measurement Noise We Can Nearly Estimate Accelerometer Errors Perfectly



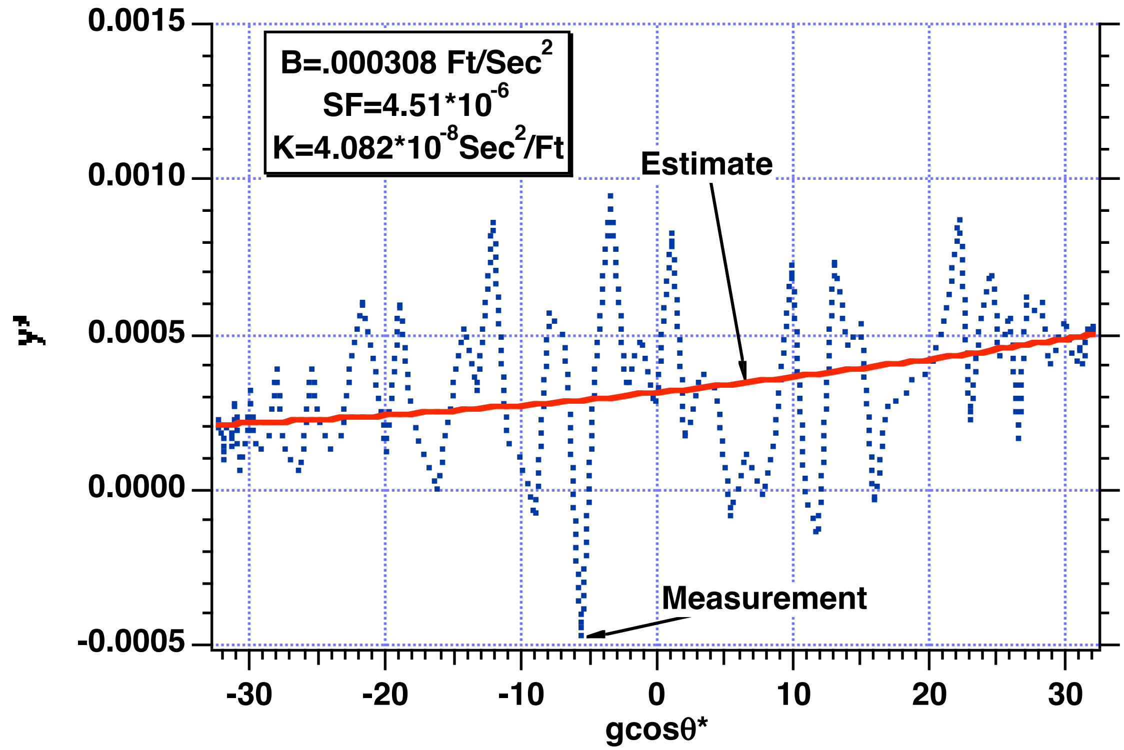
Truth

B = .000322 ft/sec^2

SF = 5*10^{-6}

K = 3.106*10^{-8} sec^2/ft

There is a Difference Between Truth and Estimates With 10 μR of Measurement Noise



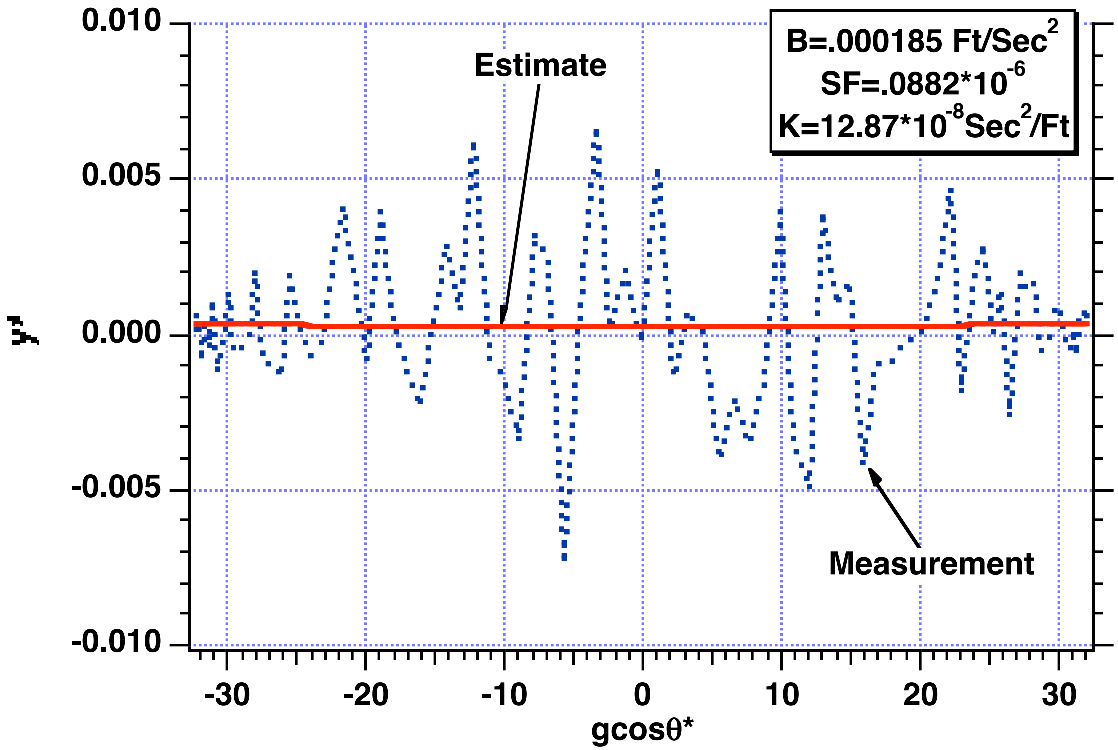
Truth

$B = .000322$ ft/sec^2

$SF = 5*10^{-6}$

$K = 3.106*10^{-8}$ sec^2/ft

With 100 μR of Measurement Noise We Can Not Estimate Bias, Scale Factor Errors or G-Sensitive Drift



Truth

B = .000322 ft/sec^2

SF = 5*10^{-6}

K = 3.106*10^{-8} sec^2/ft

Method of Least Squares
Summary

- Method of least squares is a batch processing method
  - All data has to be collected before estimates can be made
- Best to use filter order that is matched to signal order
  - If filter order is too low get divergence
  - If filter order is too high may be fitting noise rather than signal
- Batch processing formulas for various order least squares filters presented