Kalman Filters in a Non Polynomial World

Kalman Filters in a Non Polynomial World
Overview

- Tracking a sinusoid with a polynomial Kalman filter
    - First and second-order
    - With and without process noise
- Special purpose Kalman filter for tracking a sinusoid
- When is it important to have accurate fundamental matrix
- Suspension system example

Tracking a Sinusoid With a Polynomial Kalman Filter

Tracking a Sinusoid With First-Order Polynomial Kalman Filter-1

Measurement

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

The true signal and it's derivative are

$$x = \sin\omega t$$

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

For a first-order polynomial Kalman filter we assume

$$\ddot{x} = u_s$$

This is a general model and not perfect for this example

In state space form our imperfect model of the real world and measurement equation are

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ u_s \end{bmatrix}$$ ← **Polynomial filter formulation**

$$x^* = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + v$$

Therefore the fundamental and measurement noise matrices are

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

Tracking a Sinusoid With First-Order Polynomial Kalman Filter-2

Recall Kalman filtering equation is given by

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

Substitution and simplification yields first-order polynomial Kalman filter

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

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

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

Gains obtained from Riccati equations

$$\mathbf{M_k} = \mathbf{\Phi_k}\mathbf{P}_{\mathbf{k}-1}\mathbf{\Phi_k^T} + \mathbf{Q_k}$$

$$\mathbf{K_k} = \mathbf{M_k}\mathbf{H^T}(\mathbf{H}\mathbf{M_k}\mathbf{H^T} + \mathbf{R_k})^{-1}$$

$$\mathbf{P_k} = (\mathbf{I} - \mathbf{K_k}\mathbf{H})\mathbf{M_k}$$

$$\mathbf{R_k} = \sigma_n^2 \qquad \mathbf{Q_k} = \Phi_s\begin{bmatrix} \frac{T_s^3}{3} & \frac{T_s^2}{2} \\ \frac{T_s^2}{2} & T_s \end{bmatrix}$$

FORTRAN Version of First-Order Polynomial Kalman Filter and Sinusoidal Measurement-1

```
      GLOBAL DEFINE
              INCLUDE 'quickdraw.inc'
      END
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 P(2,2),Q(2,2),M(2,2),PHI(2,2),HMAT(1,2),HT(2,1),PHIT(2,2)
      REAL*8 RMAT(1,1),IDN(2,2),PHIP(2,2),PHIPPHIT(2,2),HM(1,2)
      REAL*8 HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),MHT(2,1),K(2,1)
      REAL*8 KH(2,2),IKH(2,2)
      INTEGER ORDER
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      ORDER=2
      PHIS=0.
      TS=.1
      XH=0.
      XDH=0.
      SIGNOISE=1.
      DO 14 I=1,ORDER
      DO 14 J=1,ORDER
      PHI(I,J)=0.
      P(I,J)=0.
      Q(I,J)=0.
      IDN(I,J)=0.
14    CONTINUE
      RMAT(1,1)=SIGNOISE**2
      IDN(1,1)=1.
      IDN(2,2)=1.
      P(1,1)=99999999999.
      P(2,2)=99999999999.
      PHI(1,1)=1
      PHI(1,2)=TS
      PHI(2,2)=1
      HMAT(1,1)=1.
      HMAT(1,2)=0.
      CALL MATTRN(PHI,ORDER,ORDER,PHIT)
      CALL MATTRN(HMAT,1,ORDER,HT)
      Q(1,1)=PHIS*TS**3/3
      Q(1,2)=PHIS*TS*TS/2
      Q(2,1)=Q(1,2)
      Q(2,2)=PHIS*TS
```

Initialize many matrices to zero (DO 14 loop through 14 CONTINUE)

Identity, initial covariance, fundamental and measurement matrices (IDN(1,1)=1. through HMAT(1,2)=0.)

Process noise matrix (Q(1,1) through Q(2,2))

FORTRAN Version of First-Order Polynomial Kalman Filter and Sinusoidal Measurement-2

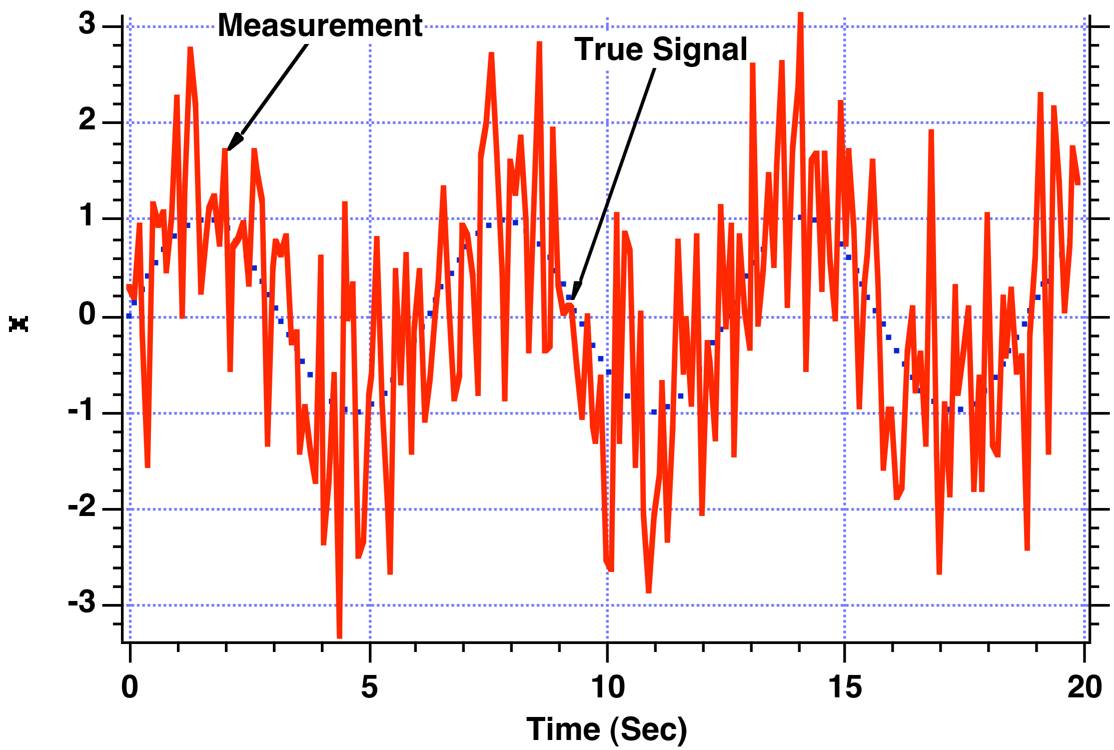
```
      DO 10 T=0.,20.,TS
            CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
            CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,PHIPPHIT)
            CALL MATADD(PHIPPHIT,ORDER,ORDER,Q,M)
            CALL MATMUL(HMAT,1,ORDER,M,ORDER,ORDER,HM)
            CALL MATMUL(HM,1,ORDER,HT,ORDER,1,HMHT)
            CALL MATADD(HMHT,ORDER,ORDER,RMAT,HMHTR)
            HMHTRINV(1,1)=1./HMHTR(1,1)
            CALL MATMUL(M,ORDER,ORDER,HT,ORDER,1,MHT)
            CALL MATMUL(MHT,ORDER,1,HMHTRINV,1,1,K)
            CALL MATMUL(K,ORDER,1,HMAT,1,ORDER,KH)
            CALL MATSUB(IDN,ORDER,ORDER,KH,IKH)
            CALL MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
            CALL GAUSS(XNOISE,SIGNOISE)
            X=SIN(T)
            XD=COS(T)
            XS=X+XNOISE
            RES=XS-XH-TS*XDH
            XH=XH+XDH*TS+K(1,1)*RES
            XDH=XDH+K(2,1)*RES
            WRITE(9,*)T,X,XS,XH,XD,XDH
            WRITE(1,*)T,X,XS,XH,XD,XDH
10    CONTINUE
      PAUSE
      CLOSE(1)
      END
```

Riccati equations (CALL MATMUL(PHI,...) through CALL MATMUL(IKH,...,P))

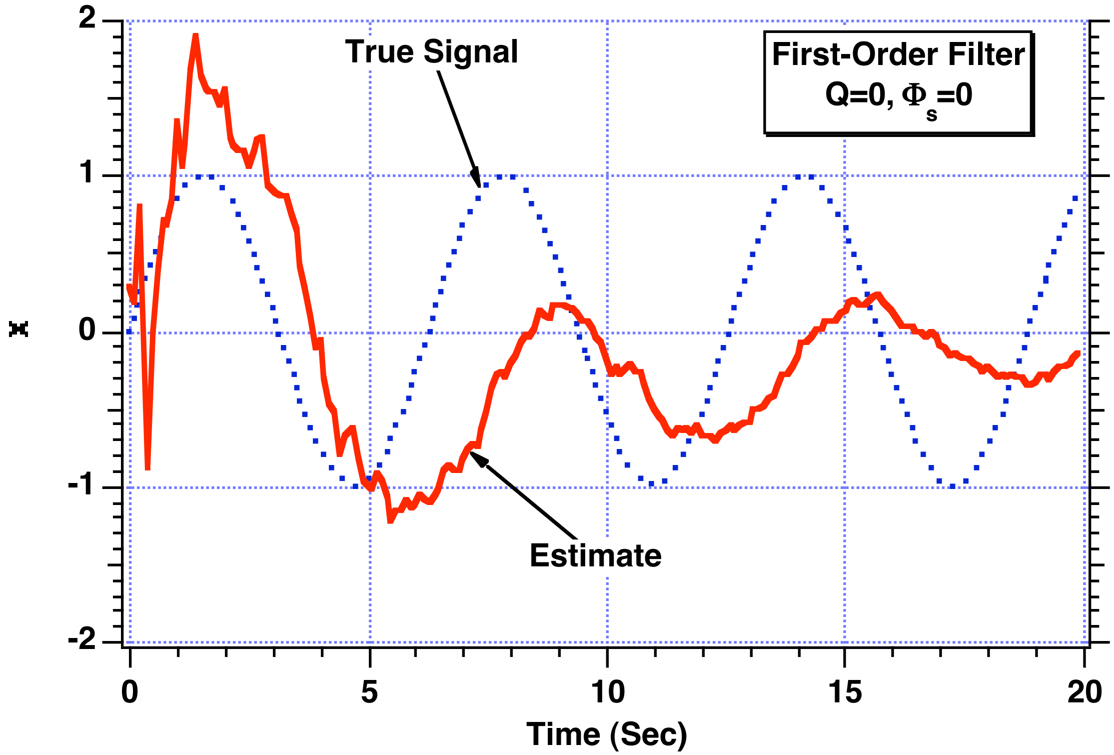
First-order polynomial Kalman filter (RES=..., XH=..., XDH=...)

Write data to screen and file (WRITE statements)

Sinusoidal Measurement is Very Noisy



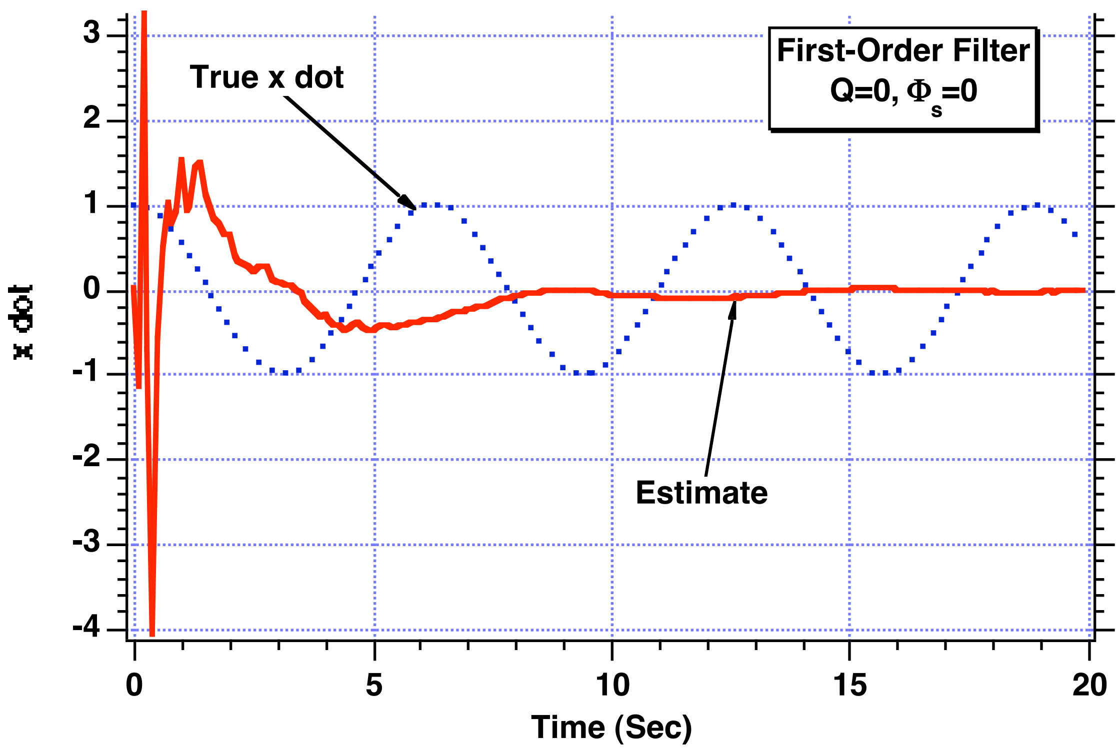
First-Order Polynomial Kalman Filter Has Difficulty in Tracking the Sinusoidal Signal



***With zero process noise filter believes real world signal is ramp**

$$\ddot{x} = 0$$

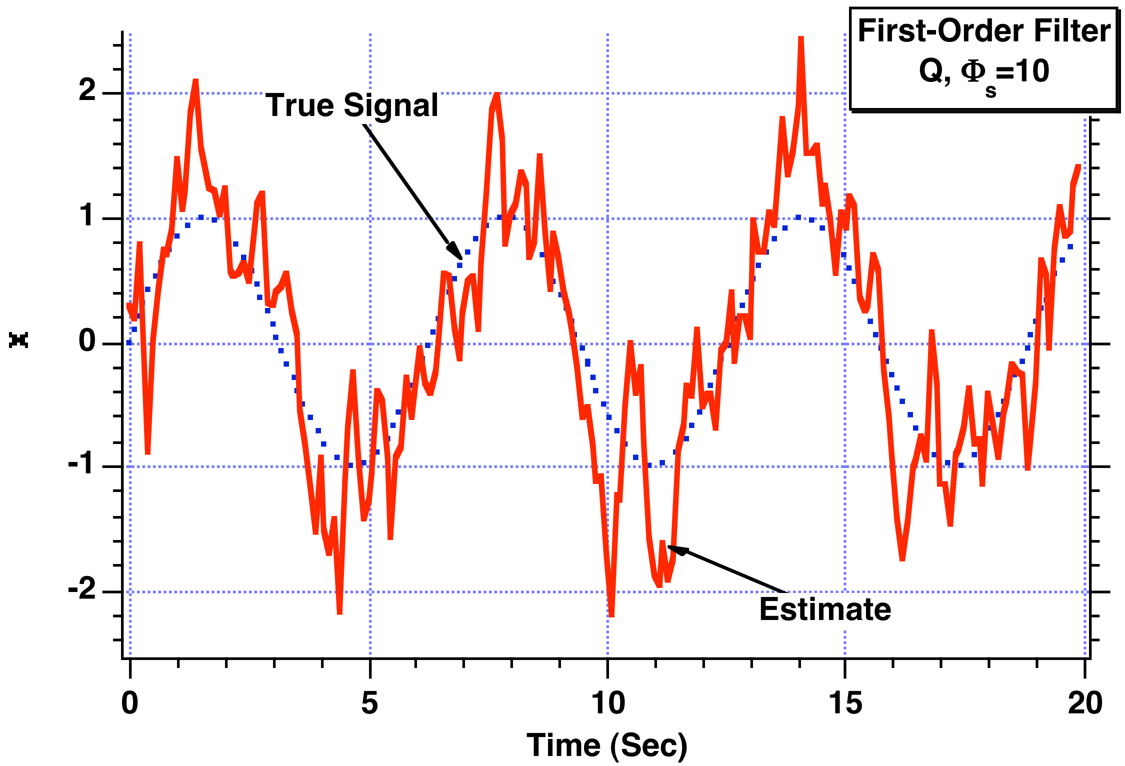
First-Order Polynomial Kalman Filter Does Poorly in Estimating the Derivative of the True Signal



***With zero process noise filter believes real world derivative is constant**

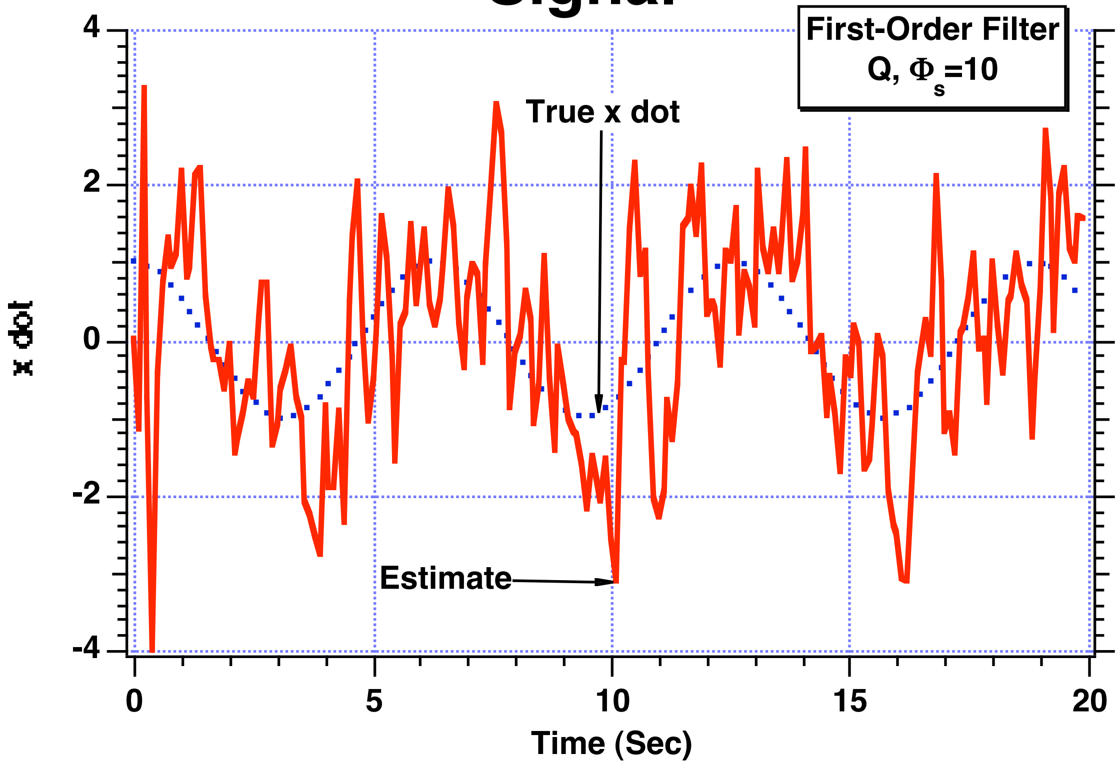
$$\ddot{x} = 0$$

Adding Process Noise Yields Better Tracking at Expense of Noisier Estimate



$$\ddot{x} = u_s$$

First-Order Filter With Process Noise is Now Able to Provide Noisy Estimate of Derivative of True Signal



$\ddot{x} = u_s$

Tracking a Sinusoid With Second-Order Polynomial Kalman Filter-1

Measurement

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

The true signal and it's derivatives are

$$x = \sin\omega t$$

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

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

For a first-order polynomial Kalman filter we assume

$$\dddot{x} = u_s$$

This again is a general model and is not perfect for this example

In state space form our model of the real world is

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

Polynomial filter formulation

Measurement model

$$x^* = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + v$$

Tracking a Sinusoid With Second-Order Polynomial Kalman Filter-2

Therefore the fundamental and measurement noise matrices are

$$\mathbf{\Phi_k} = \begin{bmatrix} 1 & T_s & .5T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{H} = [1 \quad 0 \quad 0]$$

Recall Kalman filtering equation is given by

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

Substitution and simplification yields second-order polynomial Kalman filter

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

$$\hat{x}_k = \hat{x}_{k-1} + T_s\hat{\dot{x}}_{k-1} + .5T_s^2\hat{\ddot{x}}_{k-1} + K_{1_k}RES_k$$

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

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

Gains obtained from Riccati equations

$$\mathbf{M_k} = \mathbf{\Phi_k}\mathbf{P}_{\mathbf{k}-1}\mathbf{\Phi_k^T} + \mathbf{Q_k}$$

$$\mathbf{K_k} = \mathbf{M_k}\mathbf{H^T}(\mathbf{H}\mathbf{M_k}\mathbf{H^T} + \mathbf{R_k})^{-1}$$

$$\mathbf{P_k} = (\mathbf{I} - \mathbf{K_k}\mathbf{H})\mathbf{M_k}$$

$$\mathbf{R_k} = \sigma_n^2$$

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

MATLAB Version of Second-Order Polynomial Kalman Filter and Sinusoidal Measurement-1

```
ORDER =3;
PHIS=0.;
TS=.1;
XH=0.;              Initial state estimates
XDH=0.;
XDDH=0;
SIGNOISE=1.;
PHI=[1 TS .5*TS*TS;0 1 TS;0 0 1];           Fundamental and initial covariance matrices
P=[99999999 0 0;0 999999999 0;0 0 999999999];
IDNP=eye(ORDER);
Q=zeros(ORDER);
RMAT=SIGNOISE^2;
HMAT=[1 0 0];
HT=HMAT';
PHIT=PHI';
Q(1,1)=PHIS*TS^5/20;
Q(1,2)=PHIS*TS^4/8;
Q(1,3)=PHIS*TS^3/6;
Q(2,1)=Q(1,2);
Q(2,2)=PHIS*TS^3/3;       Process noise matrix
Q(2,3)=PHIS*TS*TS/2;
Q(3,1)=Q(1,3);
Q(3,2)=Q(2,3);
Q(3,3)=PHIS*TS;
count=0;
for T=0:TS:20
        PHIP=PHI*P;
        PHIPPHIT=PHIP*PHIT;
        M=PHIPPHIT+Q;
        HM=HMAT*M;
        HMHT=HM*HT;              Riccati equations
        HMHTR=HMHT+RMAT;
        HMHTRINV=inv(HMHTR);
        MHT=M*HT;
        K=MHT*HMHTRINV;
        KH=K*HMAT;
        IKH=IDNP-KH;
        P=IKH*M;
```

MATLAB Version of Second-Order Polynomial Kalman Filter and Sinusoidal Measurement-2

```
        XNOISE=SIGNOISE*randn;
        X=sin(T);
        XD=cos(T);
        XDD=-sin(T);
        XS=X+XNOISE;
        RES=XS-XH-TS*XDH-.5*TS*TS*XDDH;
        XH=XH+XDH*TS+.5*TS*TS*XDDH+K(1,1)*RES;
        XDH=XDH+XDDH*TS+K(2,1)*RES;
        XDDH=XDDH+K(3,1)*RES;
        count=count+1;
        ArrayT(count)=T;
        ArrayX(count)=X;
        ArrayXS(count)=XS;
        ArrayXH(count)=XH;
        ArrayXD(count)=XD;
        ArrayXDH(count)=XDH;
        ArrayXDD(count)=XDD;
        ArrayXDDH(count)=XDDH;
end
figure
plot(ArrayT,ArrayX,ArrayT,ArrayXH),grid
xlabel('Time (Sec)')
ylabel('Estimate and True Signal')
axis([0 20 -2 2])
figure
plot(ArrayT,ArrayXD,ArrayT,ArrayXDH),grid
xlabel('Time (Sec)')
ylabel('XD and XDH')
axis([0 20 -4 4])
clc
output=[ArrayT',ArrayX',ArrayXH',ArrayXD',ArrayXDH',ArrayXDD',ArrayXDDH'];
save datfil output  -ascii
disp 'simulation finished'
```

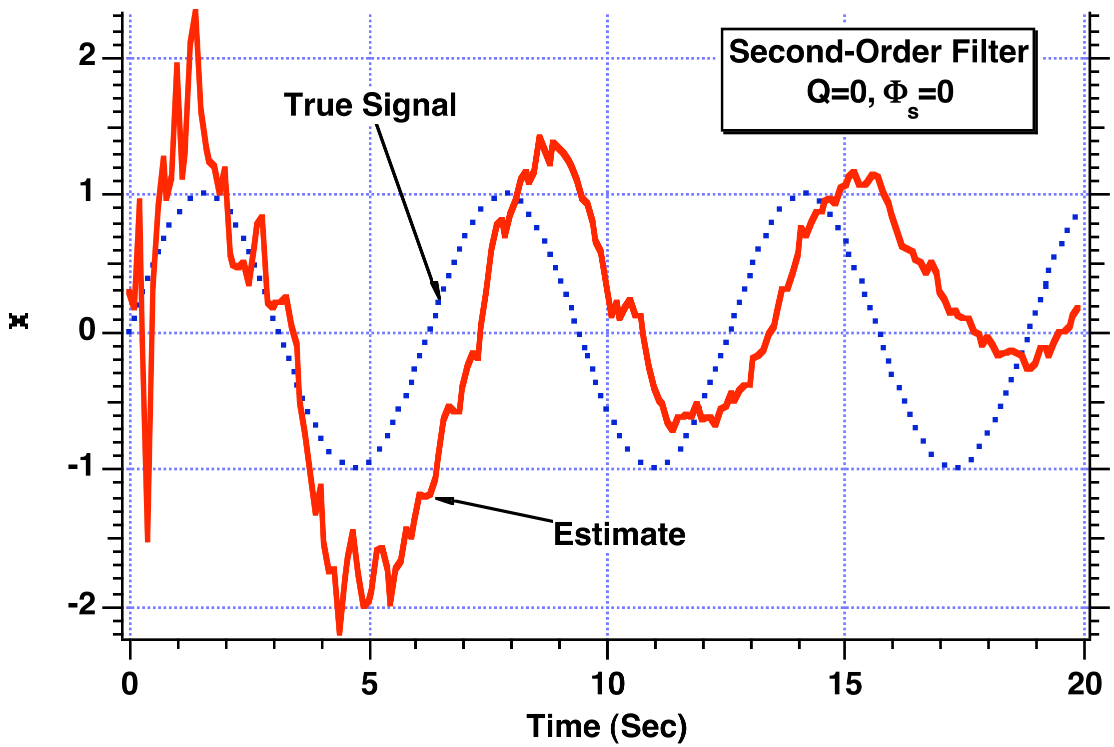
Second-order polynomial Kalman filter (RES ... XDDH lines)

Store data as arrays (count ... ArrayXDDH lines)

Plot some results (figure ... axis lines)

Write some results to a file (output ... save lines)

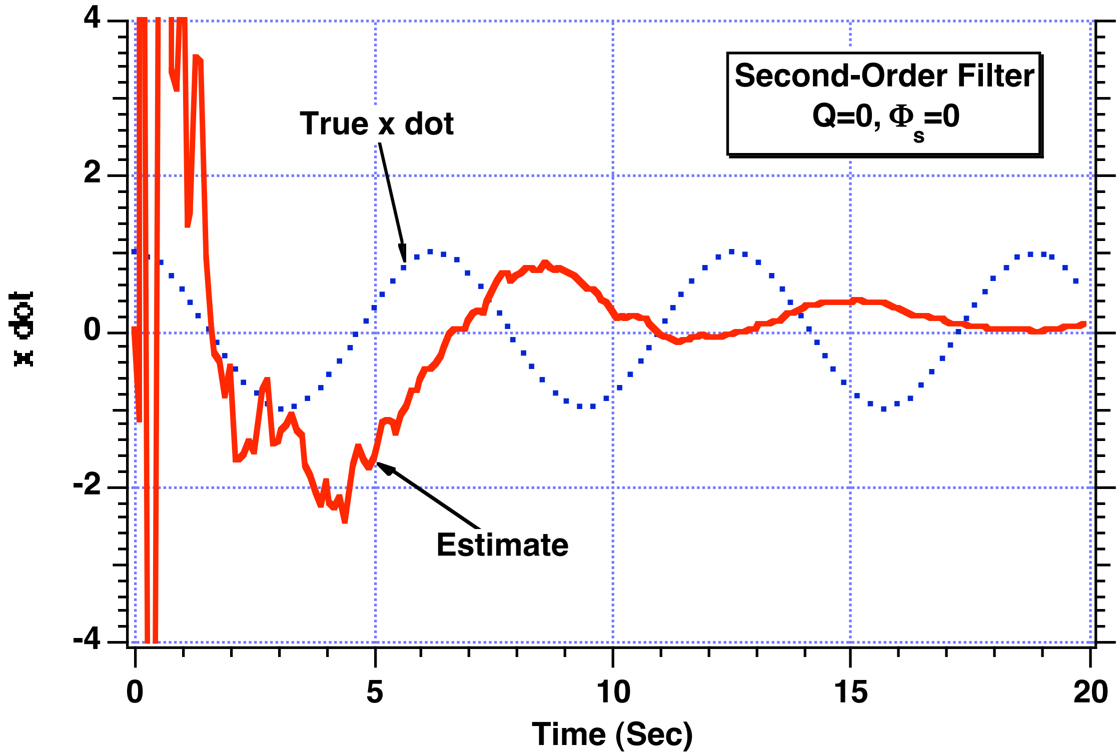
Higher Order Polynomial Kalman Filter With Zero Process Noise Yields Better but Noisier Estimates



***With zero process noise filter believes real world signal is parabola**

$$\dddot{x} = 0$$

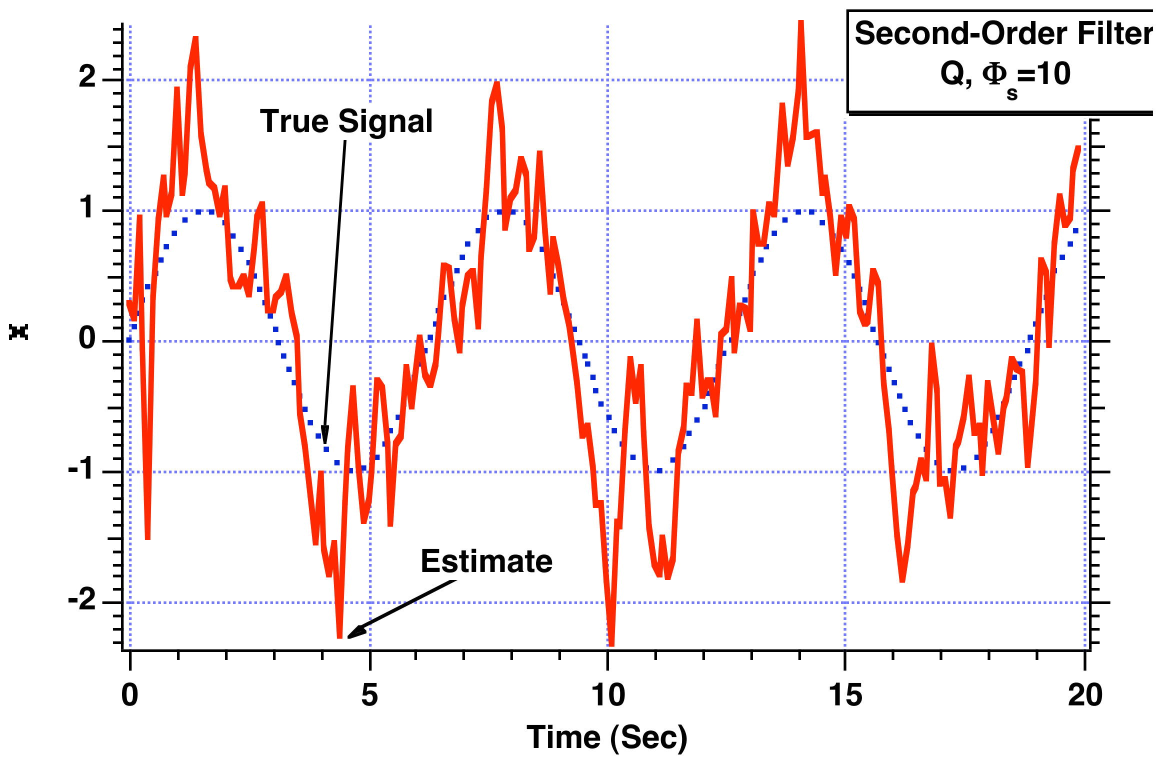
Higher Order Polynomial Kalman Filter Does Better Job of Tracking Derivative of True Signal



***With zero process noise filter believes real world derivative is ramp**

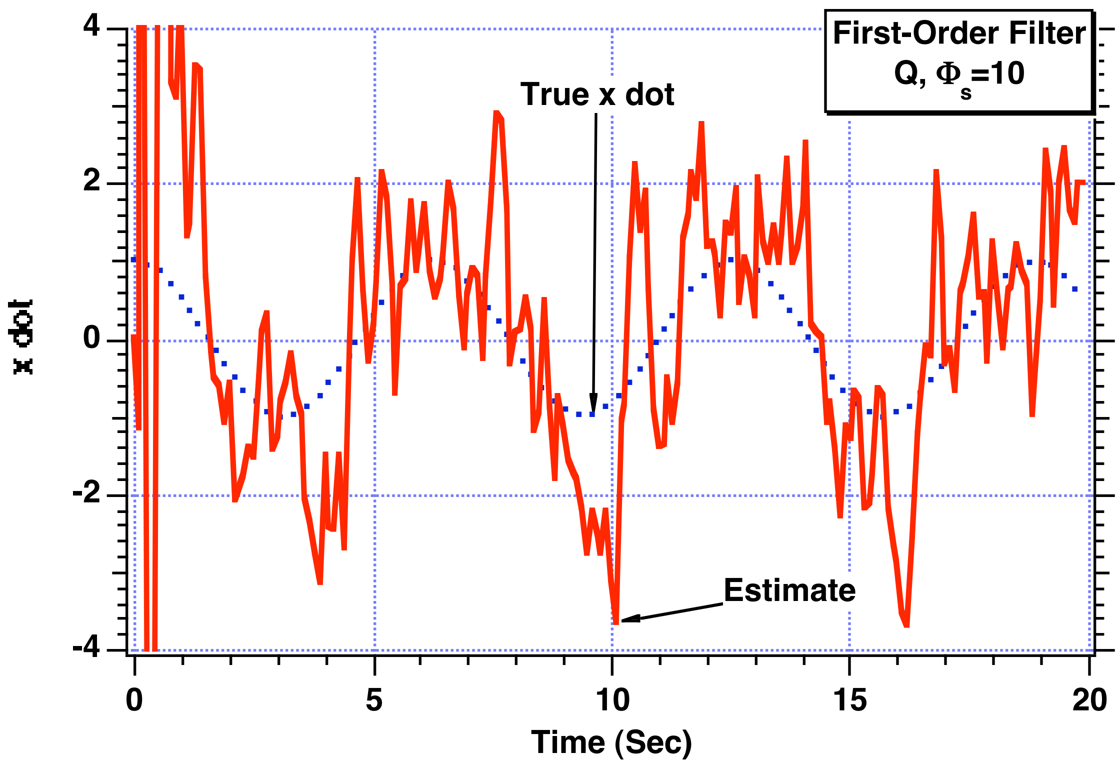
$$\dddot{x} = 0$$

Filter Lag Has Been Removed by the Addition of Process Noise



$$\dddot{x} = u_s$$

Estimate of Derivative Has Been Improved by the Addition of Process Noise



$$\dddot{x} = u_s$$

Can We Do Better With a Non Polynomial Kalman Filter?

Developing Model for Sinusoidal Kalman Filter

Recall that actual signal given by

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

Taking derivative twice

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

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

Or

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

In state space form

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

← **This neglects process noise since we have a perfect model**

Therefore systems dynamics matrix is

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

Measurement equation

$$x^* = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + v$$

Finding Fundamental Matrix for Sinusoidal Kalman Filter

We know that

$$\Phi(\mathbf{t}) = \pounds^{-1}[(\mathbf{sI} - \mathbf{F})^{-1}]$$

Substitution yields

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

After taking the inverse we get

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

Converting to the time domain yields

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

Or in discrete form

$$\Phi_{\mathbf{k}} = \begin{bmatrix} \cos\omega T_s & \frac{\sin\omega T_s}{\omega} \\ -\omega\sin\omega T_s & \cos\omega T_s \end{bmatrix}$$

Sinusoidal Kalman Filter Equations

Recall

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

Substitution and simplification yield

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

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

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

MATLAB Version of Sinusoidal Kalman Filter and Sinusoidal Measurement-1

```
ORDER=2;
PHIS=0.;
W=1;
A=1;
TS=.1;
XH=0.;
XDH=0.;
SIGNOISE=1.;
PHI=[cos(W*TS) sin(W*TS)/W ;-W*sin(W*TS) cos(W*TS)];
P=[99999999 0;0 999999999];
IDNP=eye(ORDER);
Q=zeros(ORDER);
RMAT=SIGNOISE^2;
HMAT=[1 0];
HT=HMAT';
PHIT=PHI';
count=0;
for T=0:TS:20
        PHIP=PHI*P;
        PHIPPHIT=PHIP*PHIT;
        M=PHIPPHIT+Q;
        HM=HMAT*M;
        HMHT=HM*HT;
        HMHTR=HMHT+RMAT;
        HMHTRINV=inv(HMHTR);
        MHT=M*HT;
        K=MHT*HMHTRINV;
        KH=K*HMAT;
        IKH=IDNP-KH;
        P=IKH*M;
```

Fundamental and initial Covariance matrices

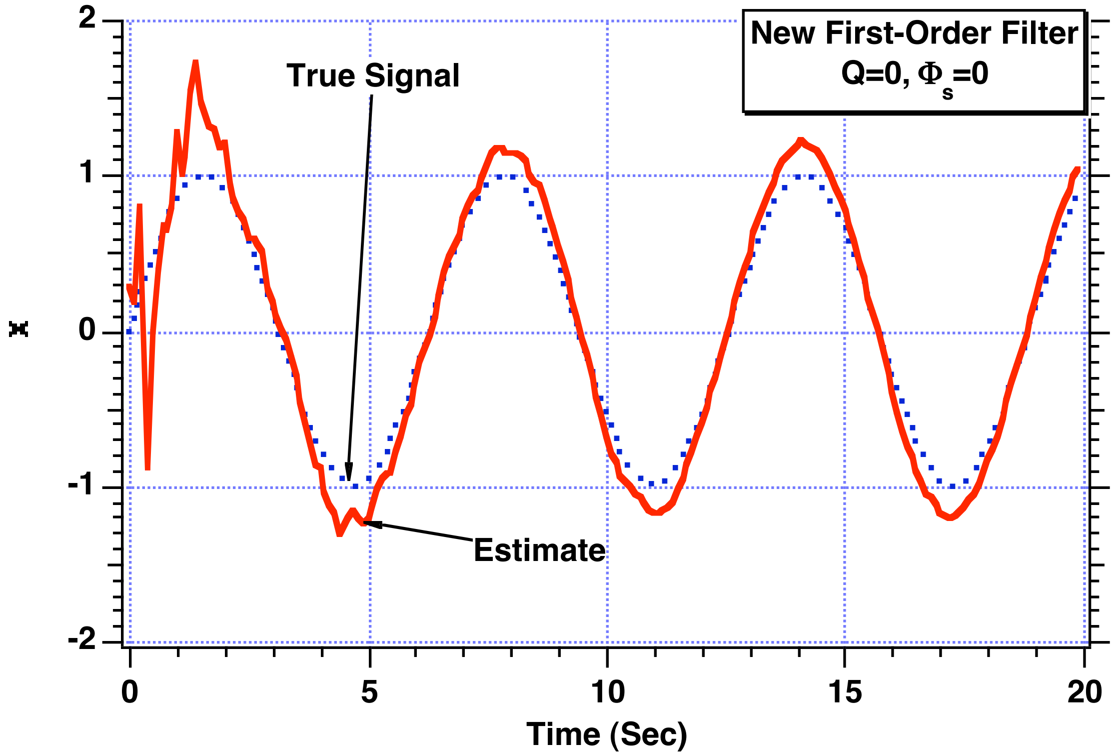
Riccati equations

MATLAB Version of Sinusoidal Kalman Filter and Sinusoidal Measurement-2

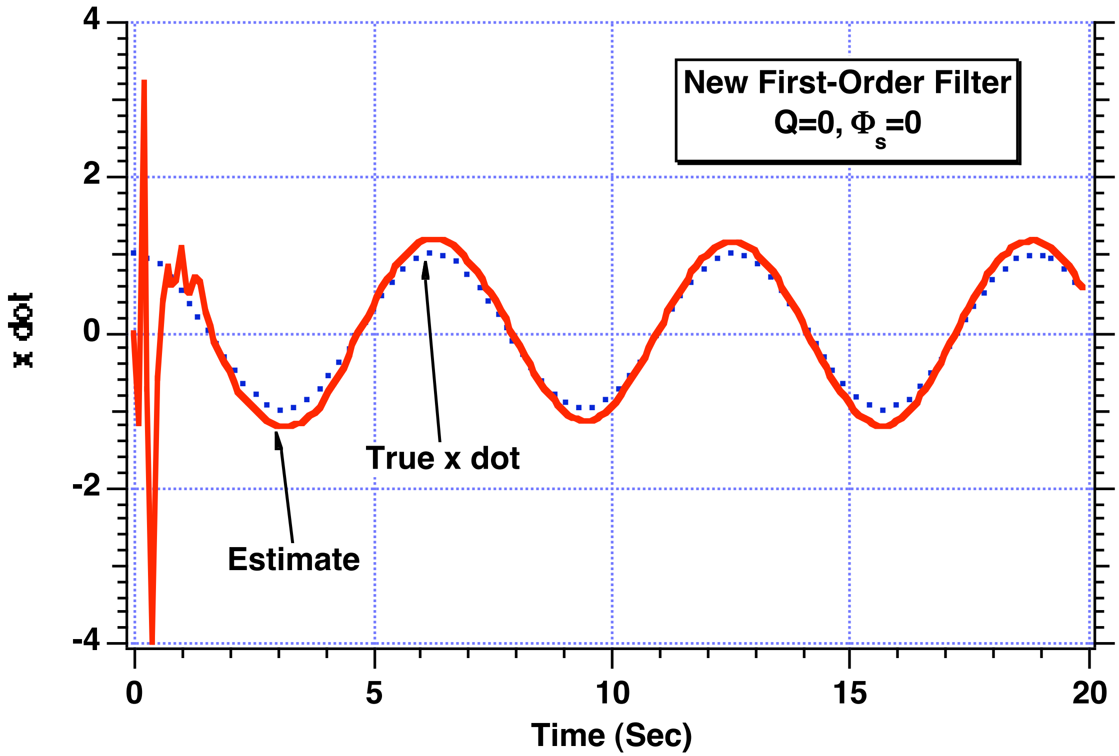
```
        XNOISE=SIGNOISE*randn;
        X=A*sin(W*T);
        XD=A*W*cos(W*T);
        XS=X+XNOISE;
        XHOLD=XH;
        RES=XS-XH*cos(W*TS)-sin(W*TS)*XDH/W;
        XH=cos(W*TS)*XH+XDH*sin(W*TS)/W+K(1,1)*RES;
        XDH=-W*sin(W*TS)*XHOLD+XDH*cos(W*TS)+K(2,1)*RES;
        count=count+1;
        ArrayT(count)=T;
        ArrayX(count)=X;
        ArrayXS(count)=XS;
        ArrayXH(count)=XH;
        ArrayXD(count)=XD;
        ArrayXDH(count)=XDH;
end
figure
plot(ArrayT,ArrayX,ArrayT,ArrayXH),grid
xlabel('Time (Sec)')
ylabel('Estimate and True Signal')
axis([0 20 -2 2])
figure
plot(ArrayT,ArrayXD,ArrayT,ArrayXDH),grid
xlabel('Time (Sec)')
ylabel('XD and XDH')
axis([0 20 -4 4])
clc
output=[ArrayT',ArrayX',ArrayXS'ArrayXDH'];
save datfil output  -ascii
output=[ArrayT',ArrayXD',ArrayXDH'];
save covfil output  -ascii
disp 'simulation finished'
```

Sinusoidal Kalman filter (lines RES= through XDH=)

New Filter Dramatically Improves Estimate of Signal



New Filter Dramatically Improves Estimate of Derivative of Signal



Where is the Fundamental Matrix Most Important?

Using an Approximate Fundamental Matrix

We would always like to use an exact fundamental matrix but sometimes that is impossible

Recall that

$$\mathbf{\Phi_k} = \mathbf{e}^{\mathbf{F}T_s} \approx \mathbf{I} + \mathbf{F}T_s + \frac{(\mathbf{F}T_s)^2}{2} + ...$$

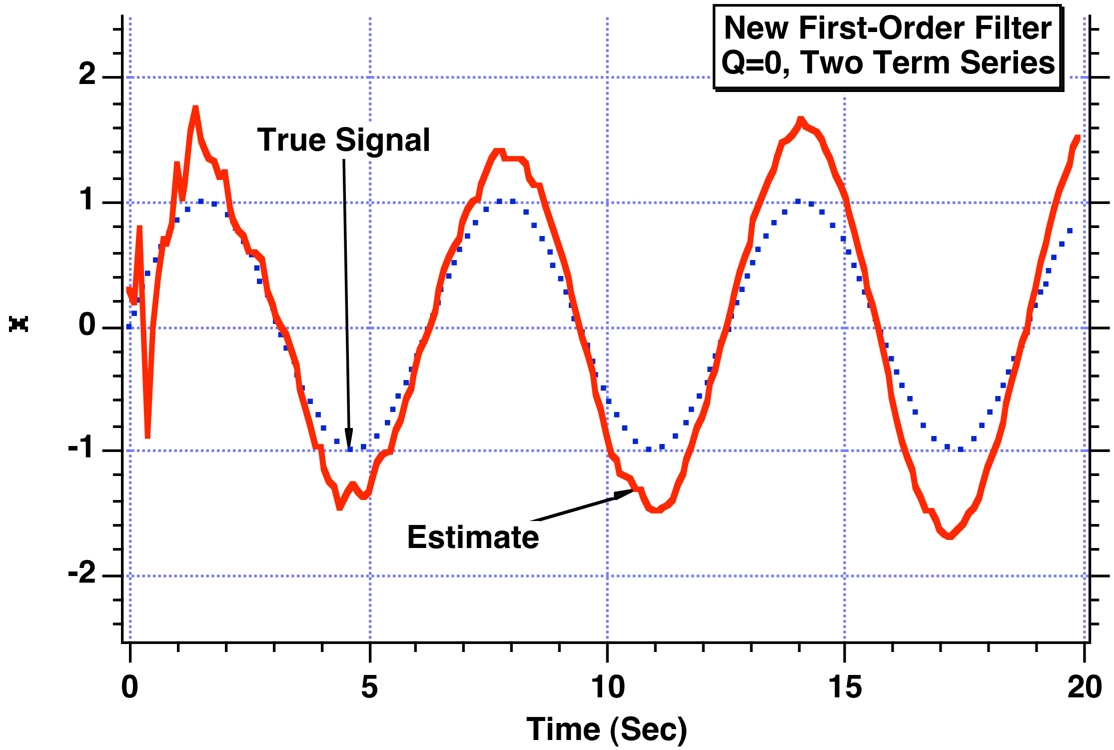
A two-term approximation yields

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

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

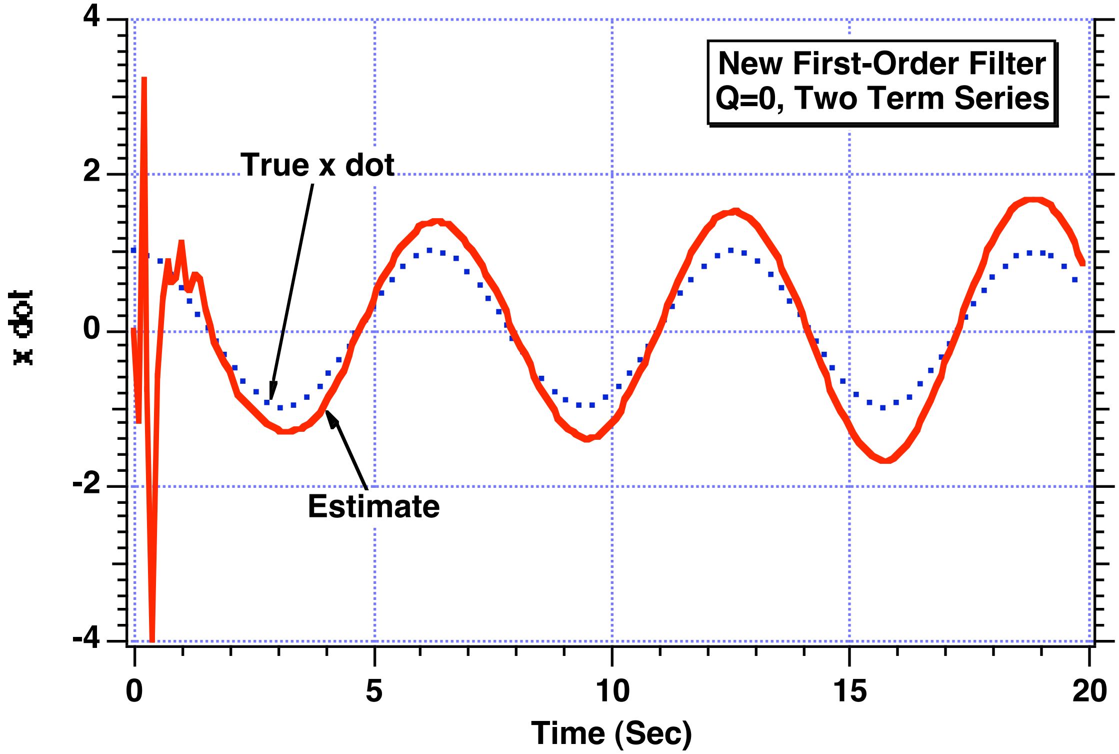
Fundamental matrix shows up in the filter and Riccati equations

Estimate of Signal is Worse When Fundamental Matrix is Approximate



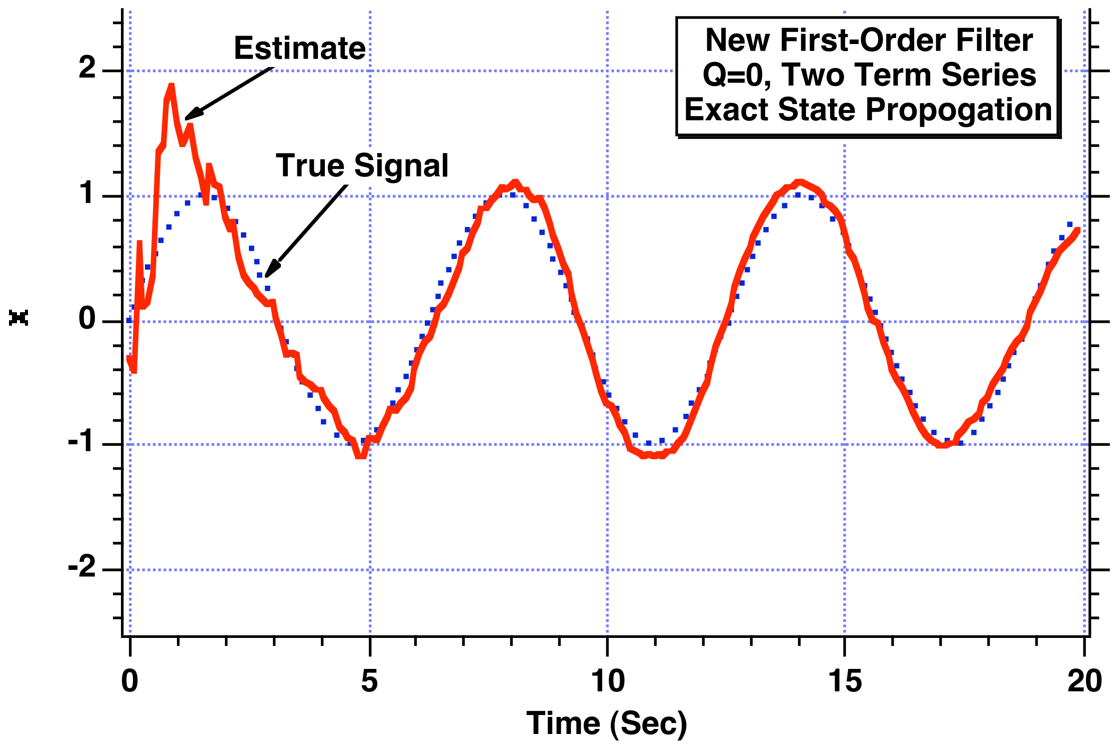
***Approximate fundamental matrix in both filter and Riccati equations**

Estimate of Signal Derivative is Also Worse When Fundamental Matrix is Approximate



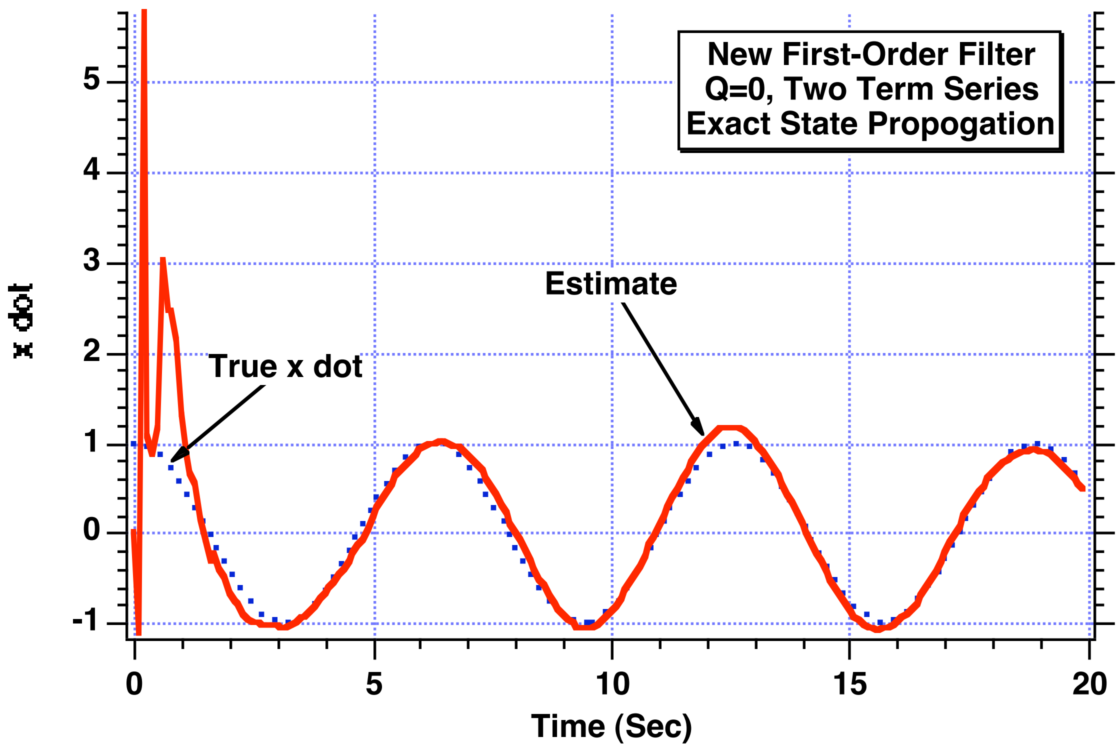
*Approximate fundamental matrix in both filter and Riccati equations

Estimate of Signal is Excellent When Approximation for Fundamental Matrix is Only Used in Riccati Equations



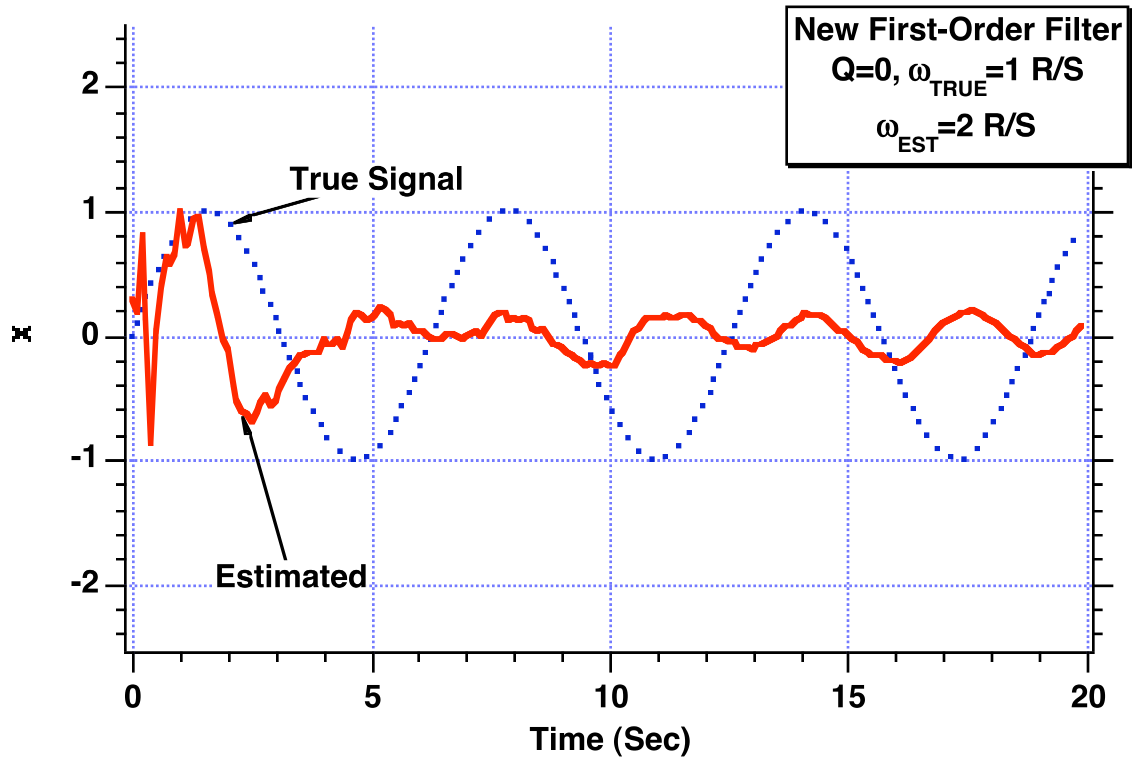
*Approximate fundamental matrix in Riccati equations
Exact fundamental matrix in filter

Estimate of Signal Derivative is Also Excellent When Approximation for Fundamental Matrix is Only Used in Riccati Equations



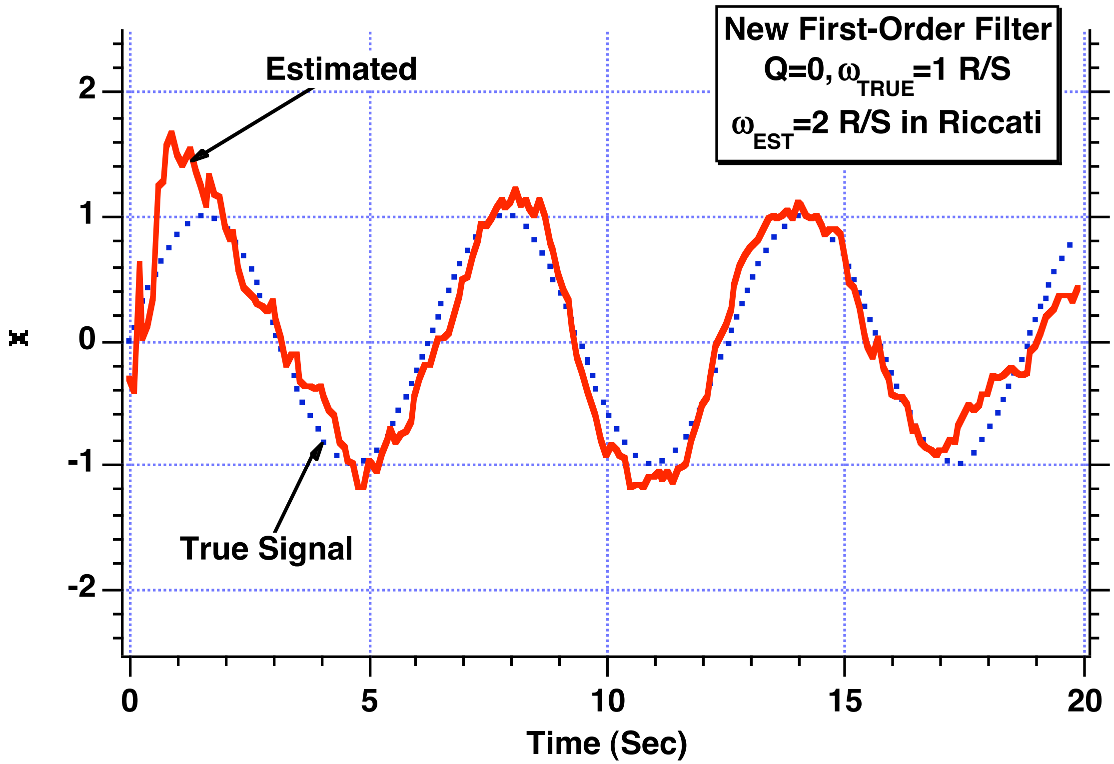
*Approximate fundamental matrix in Riccati equations
Exact fundamental matrix in filter

Kalman Filter that Depends on Knowing Signal Frequency is Sensitive to Modeling Errors



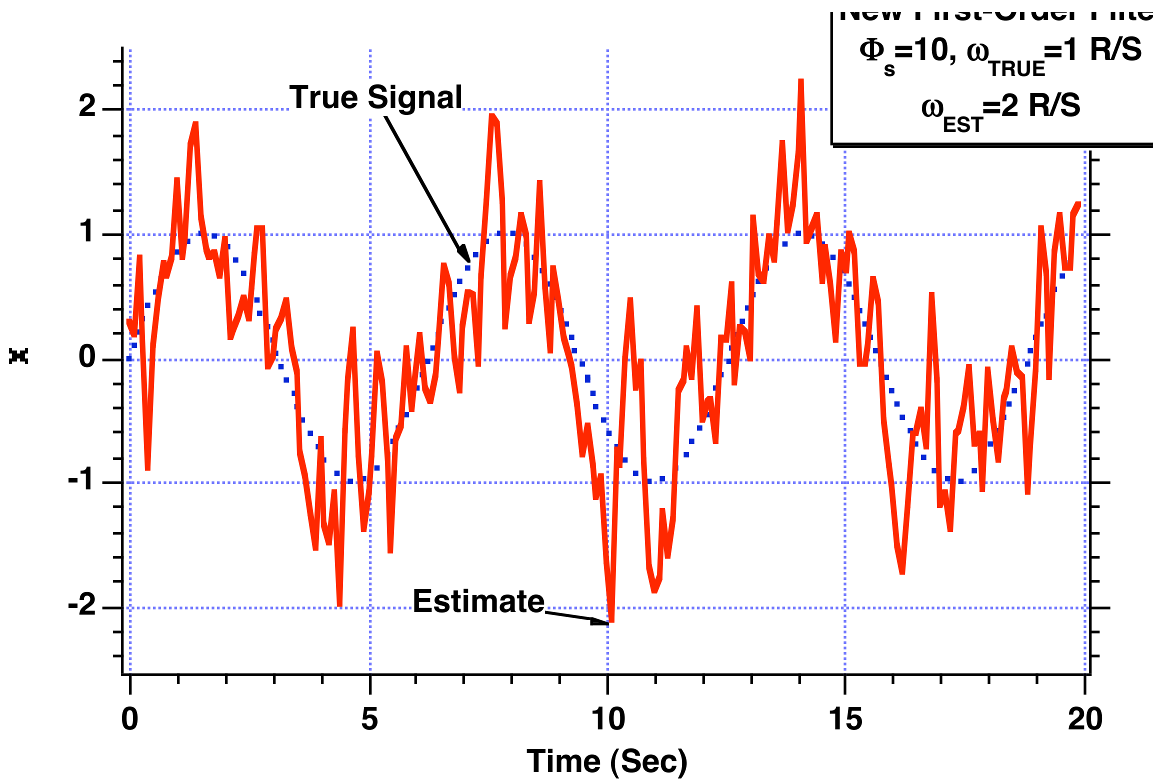
*Estimated frequency shows up in fundamental matrix
Fundamental matrix shows up in filter and Riccati equations

If Frequency Mismatch is in Riccati Equations Only, Filter Still Gives Excellent Estimates



*Inaccurate fundamental matrix in Riccati equations
Exact fundamental matrix in filter

Adding Process Noise Enables Kalman Filter With Bad A Priori Information to Provide Good Estimates



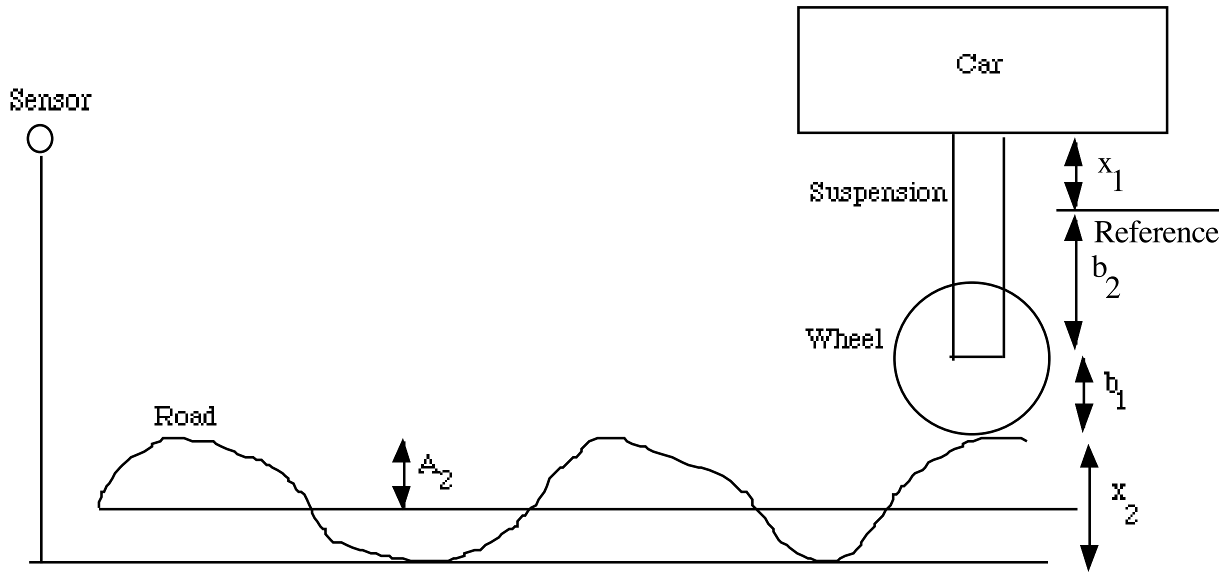
*Inaccurate fundamental matrix in both filter and Riccati equations

Fundamental Matrix Summary

- Having an exact fundamental matrix enables the filter to have good estimates
- If the fundamental matrix can not be exact
    - It is more important to somehow propagate states correctly in filter
    - Second best thing to do is add process noise

Suspension System Example

Car Riding Over a Bumpy Road



Bumpy road

$x_2 = A_2 \sin\omega t$

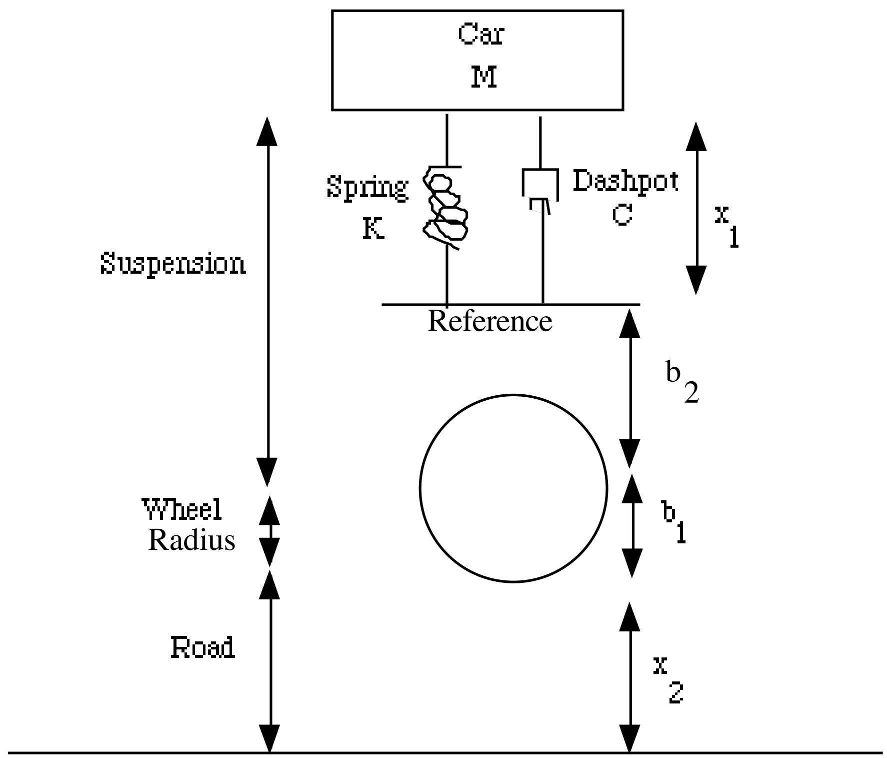
Wheel radius = b_1

Suspension length = $x_1 + b_2$

Height of car = $x_1 + b_2 + b_1 + x_2$

Sensor measures height of car

Suspension System is Represented by Spring and Dashpot



K = Spring constant

C = Coefficient of viscous damping

M = Mass of car

Developing a Model of the Real World-1

From Newton's second law

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

Dividing both sides by mass of car

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

Defining a damping and natural frequency

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

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

Substitution yields second-order differential equation

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

Since

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

Developing a Model of the Real World-2

Taking derivatives

$\dot{x}_2 = A_2 \omega \cos \omega t$

$\ddot{x}_2 = -A_2 \omega^2 \sin \omega t$ ← **This goes into differential equation**

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

Nominal Values

Bumpy road $= x_2 = A_2 \sin \omega t = .1\sin 6.28t$

Initial suspension length $= x_1 + b_2 = 1.5$ ft → $x_1 = .25$ ft, $b_2 = 1.25$ ft

Wheel radius $= b_1 = 1$ ft

$\zeta = .7$

$\omega_n = .1*6.28$

$\dot{x}_1(0) = 0$

FORTRAN Simulation That Integrates the Suspension Differential Equation

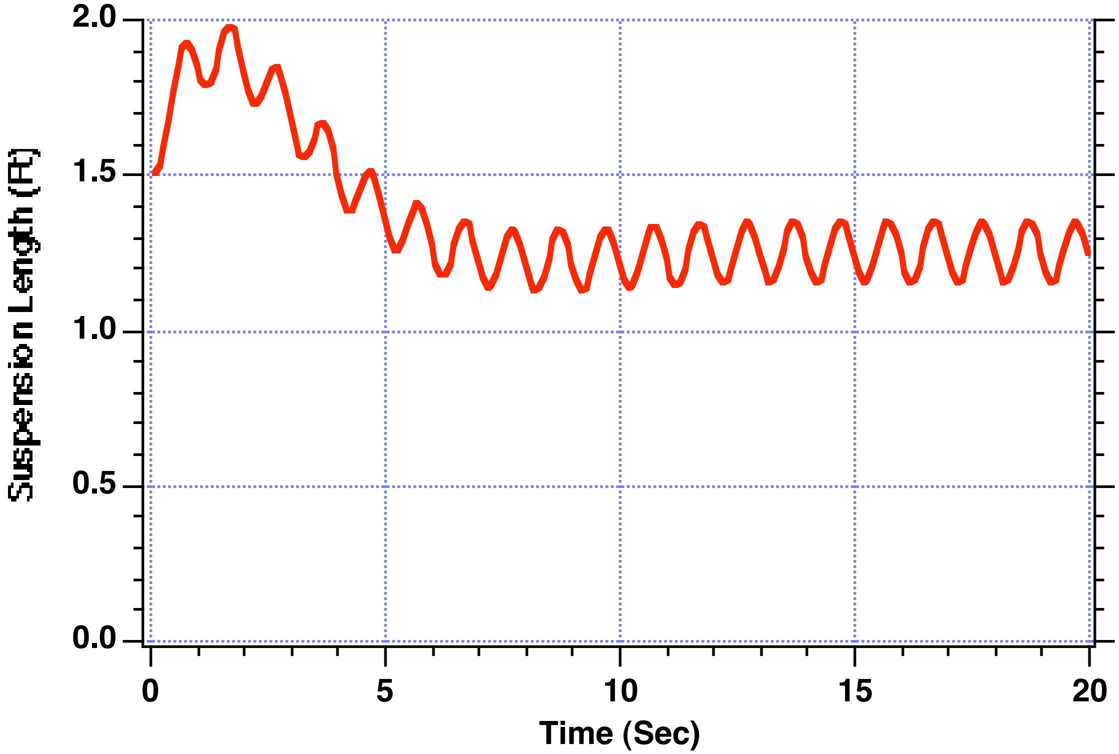
```
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
WN=6.28*.1
W=6.28*1.
Z=.7
A2=.1
X1=.25
B2=1.25
X1D=0.
B1=1.
T=0.
S=0.
H=.001
WHILE(T<=20.)
        S=S+H
        X1OLD=X1
        X1DOLD=X1D
        X2=A2*SIN(W*T)
        X2D=A2*W*COS(W*T)
        X2DD=-A2*W*W*SIN(W*T)
        X1DD=-2.*Z*WN*X1D-WN*WN*X1-X2DD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

Nominal values and initial conditions (WN=6.28*.1 through B1=1.)

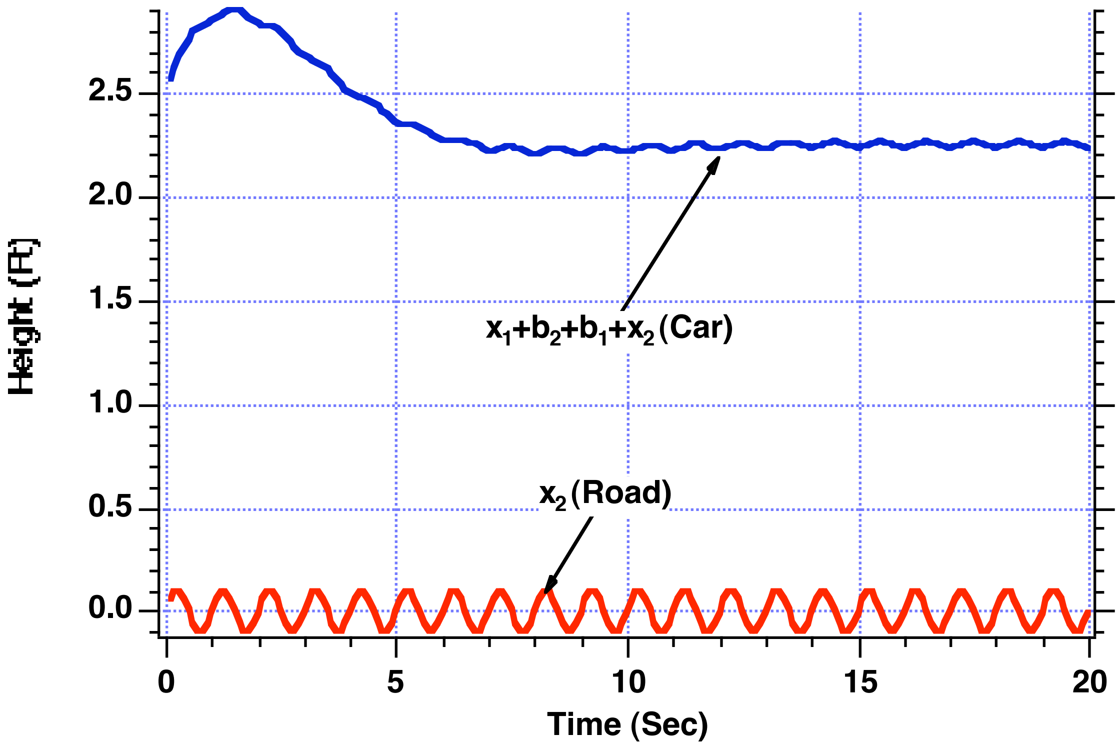
Second-Order Runge Kutta integration (S=S+H through X1D=.5*(X1DOLD+X1D+H*X1DD))

Writing data to screen and file (IF(S>=.09999)THEN block)

Suspension Oscillates at the Road Frequency



Suspension Enables the Car to Have a Smooth Ride



Deriving Appropriate Matrices for Kalman Filter in Suspension System Example

First put model of real world in state space form

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

True sensor measurement

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

Modified measurement to fit filter formulation

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

Measurement and systems dynamics matrices given by

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta\omega_n \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} 0 & 0 \\ 0 & \Phi_s \end{bmatrix}$$

Deriving Fundamental Matrix-1

Fundamental matrix can be derived using Laplace transforms

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

Taking the inverse yields

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

If we define

$$a = -\zeta\omega_n$$

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

Then denominator becomes

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

And we get

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

Deriving Fundamental Matrix-2

From Laplace transform tables we know that

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

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

After some algebra we get

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

The discrete form can be written by inspection as

$$\boldsymbol{\Phi}_{\mathbf{k}} = \begin{bmatrix} \frac{e^{aT_s}(-a\sin bT_s + b\cos bT_s)}{b} & \frac{e^{aT_s}\sin bT_s}{b} \\ \frac{-\omega_n^2 e^{aT_s}\sin bT_s}{b} & \frac{e^{aT_s}(a\sin bT_s + b\cos bT_s)}{b} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix}$$

Finding G Matrix-1

Recall

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

Therefore

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

Discrete form of G

$$\mathbf{G_k} = \int_0^{T_s} \mathbf{\Phi}(\tau)\mathbf{G}\,d\tau$$ ← **True if input constant between sampling instants**

***Not good approximation here**

Substitution yields

$$\mathbf{G_k} = \int_0^{T_s} \begin{bmatrix} \frac{e^{a\tau}(-a\sin b\tau + b\cos b\tau)}{b} & \frac{e^{a\tau}\sin b\tau}{b} \\ \frac{-\omega_n^2 e^{a\tau}\sin b\tau}{b} & \frac{e^{a\tau}(a\sin b\tau + b\cos b\tau)}{b} \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} d\tau$$

Which simplifies to

$$\mathbf{G_k} = \int_0^{T_s} \begin{bmatrix} -\frac{e^{a\tau}\sin b\tau}{b} \\ -\frac{e^{a\tau}(a\sin b\tau + b\cos b\tau)}{b} \end{bmatrix} d\tau$$

Finding G Matrix-2

From previous slide

$$\mathbf{G_k} = \int_0^{T_s} \begin{bmatrix} -\dfrac{e^{a\tau}\sin b\tau}{b} \\ -\dfrac{e^{a\tau}(a\sin b\tau + b\cos b\tau)}{b} \end{bmatrix} d\tau$$

We know from integration tables

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

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

Therefore discrete G matrix becomes

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

Deriving Kalman Filter For Suspension Example

Kalman filter equation

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

Substitution yields

$$\begin{bmatrix} \hat{x}_{1_k} \\ \hat{\dot{x}}_{1_k} \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \Phi_{12} \\ \Phi_{21} & \Phi_{22} \end{bmatrix} \begin{bmatrix} \hat{x}_{1_{k-1}} \\ \hat{\dot{x}}_{1_{k-1}} \end{bmatrix} + \begin{bmatrix} G_1 \\ G_2 \end{bmatrix} \ddot{x}_2 +$$

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

Or in scalar form

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

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

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

$$\Phi_{11} = \frac{e^{aT_s}(-a\sin bT_s + b\cos bT_s)}{b}$$

$$\Phi_{12} = \frac{e^{aT_s}\sin bT_s}{b}$$

$$\Phi_{21} = \frac{-\omega_n^2 e^{aT_s}\sin bT_s}{b}$$

$$\Phi_{22} = \frac{e^{aT_s}(a\sin bT_s + b\cos bT_s)}{b}$$

True BASIC Version of Kalman Filter For Suspension Example-1

```
OPTION NOLET
REM UNSAVE "DATFIL"
REM UNSAVE "COVFIL"
OPEN #1:NAME "DATFIL",ACCESS OUTPUT,CREATE NEW, ORGANIZATION TEXT
OPEN #2:NAME "COVFIL",ACCESS OUTPUT,CREATE NEW, ORGANIZATION TEXT
SET #1: MARGIN 1000
SET #2: MARGIN 1000
DIM P(2,2),Q(2,2),M(2,2),PHI(2,2),HMAT(1,2),HT(2,1),PHIT(2,2)
DIM RMAT(1,1),IDNP(2,2),PHIP(2,2),PHIPPHIT(2,2),HM(1,2)
DIM HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),MHT(2,1),K(2,1)
DIM KH(2,2),IKH(2,2),G(2,1)
ORDER=2
TF=20.
SIGX=.1
TS=.01
WN=6.28*.1
W=6.28*1.
Z=.7
A=-Z*WN
B=WN*SQR(1.-Z*Z)
A2=.1
X1=.25
B2=1.25
X1D=0.
B1=1.
T=0.
S=0.
H=.001
X2DDOLD=0.
MAT PHI=ZER(ORDER,ORDER)
MAT P=ZER(ORDER,ORDER)
MAT IDNP=IDN(ORDER,ORDER)
MAT Q=ZER(ORDER,ORDER)
PHI(1,1)=EXP(A*TS)*(-A*SIN(B*TS)+B*COS(B*TS))/B
PHI(1,2)=EXP(A*TS)*SIN(B*TS)/B
PHI(2,1)=-WN*WN*EXP(A*TS)*SIN(B*TS)/B
PHI(2,2)=EXP(A*TS)*(A*SIN(B*TS)+B*COS(B*TS))/B
```

Initialize some matrices to zero

Fundamental matrix

True BASIC Version of Kalman Filter For Suspension Example-2

```
HMAT(1,1)=1.
HMAT(1,2)=0.
G(1,1)=-(EXP(A*TS)*(A*SIN(B*TS)-B*COS(B*TS))+B)/(B*(A*A+B*B))
G(2,1)=-EXP(A*TS)*SIN(B*TS)/B
MAT PHIT=TRN(PHI)
MAT HT=TRN(HMAT)
P(1,1)=9999999.
P(2,2)=9999999.
Q(2,2)=0.
X1H=X1
X1DH=X1D
RMAT(1,1)=SIGX^2
DO WHILE T<=20
        S=S+H
        X1OLD=X1
        X1DOLD=X1D
        X2=A2*SIN(W*T)
        X2D=A2*W*COS(W*T)
        X2DD=-A2*W*W*SIN(W*T)
        X1DD=-2.*Z*WN*X1D-WN*WN*X1-X2DD
        X1=X1+H*X1D
        X1D=X1D+H*X1DD
        T=T+H
        X2=A2*SIN(W*T)
        X2D=A2*W*COS(W*T)
        X2DD=-A2*W*W*SIN(W*T)
        X1DD=-2.*Z*WN*X1D-WN*WN*X1-X2DD
        X1=.5*(X1OLD+X1+H*X1D)
        X1D=.5*(X1DOLD+X1D+H*X1DD)
        IF S>=(TS-.00001) THEN
                S=0.
                MAT PHIP=PHI*P
                MAT PHIPPHIT=PHIP*PHIT
                MAT M=PHIPPHIT+Q
                MAT HM=HMAT*M
                MAT HMHT=HM*HT
                MAT HMHTR=HMHT+RMAT
                HMHTRINV(1,1)=1./HMHTR(1,1)
                MAT MHT=M*HT
                MAT K=MHT*HMHTRINV
                MAT KH=K*HMAT
                MAT IKH=IDNP-KH
                MAT P=IKH*M
```

H & G matrices (HMAT(1,1) through G(2,1))

Initial covariance matrix (P(1,1), P(2,2))

Integrating differential equations using second-order Runge-Kutta technique (S=S+H through X1D=.5*(X1DOLD+X1D+H*X1DD))

Riccati equations (S=0. through MAT P=IKH*M)

True BASIC Version of Kalman Filter For Suspension Example-3

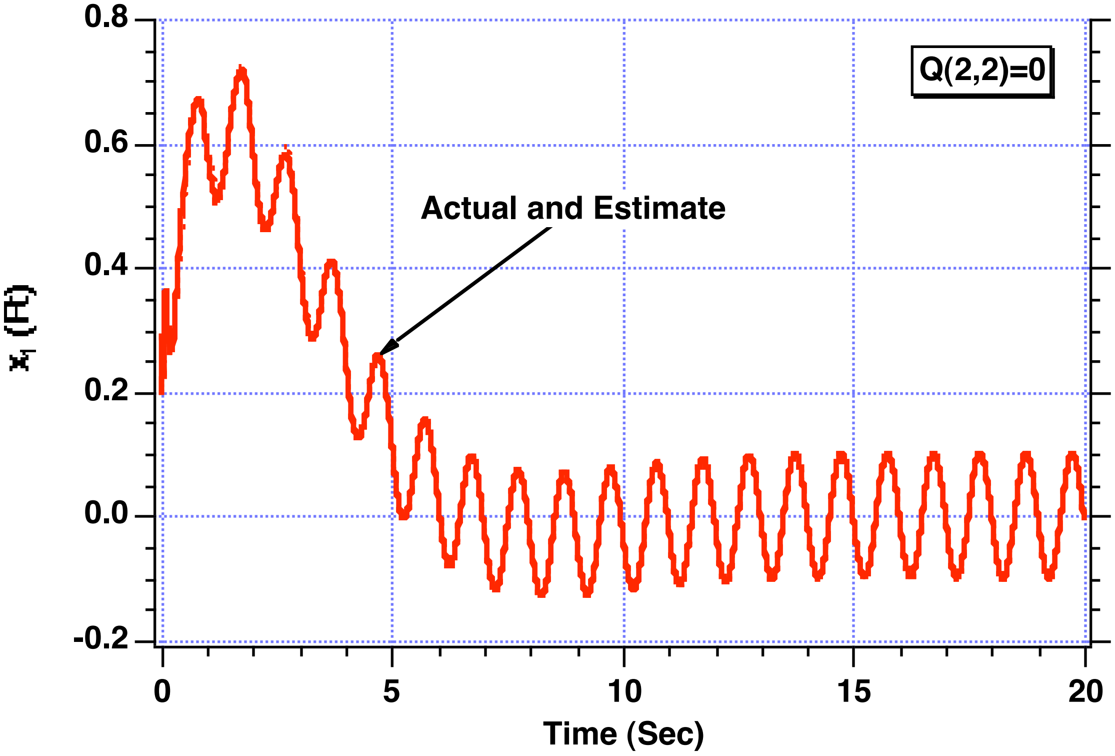
```
                    CALL GAUSS(XNOISE,SIGX)
                    XMEAS=X1+B1+X2+B2+XNOISE
                    XS=XMEAS-X2-B1-B2
                    RES=XS-PHI(1,1)*X1H-PHI(1,2)*X1DH-G(1,1)*X2DDOLD
                    X1HOLD=X1H
                    X1H=PHI(1,1)*X1H+PHI(1,2)*X1DH+G(1,1)*X2DDOLD+K(1,1)*RES

X1DH=PHI(2,1)*X1HOLD+PHI(2,2)*X1DH+G(2,1)*X2DDOLD+K(2,1)*RES
                    ERRX1=X1-X1H
                    SP11=SQR(P(1,1))
                    ERRX1D=X1D-X1DH
                    SP22=SQR(P(2,2))
                    X2DDOLD=X2DD
                    PRINT T,X1,X1H,X1D,X1DH
                    PRINT #1:T,X1,X1H,X1D,X1DH
                    PRINT #2:T,ERRX1,SP11,-SP11,ERRX1D,SP22,-SP22
          END IF
LOOP
CLOSE #1
CLOSE #2
END

SUB GAUSS(X,SIG)
LET X=RND+RND+RND+RND+RND+RND-3
LET X=1.414*X*SIG
END SUB
```

Filter

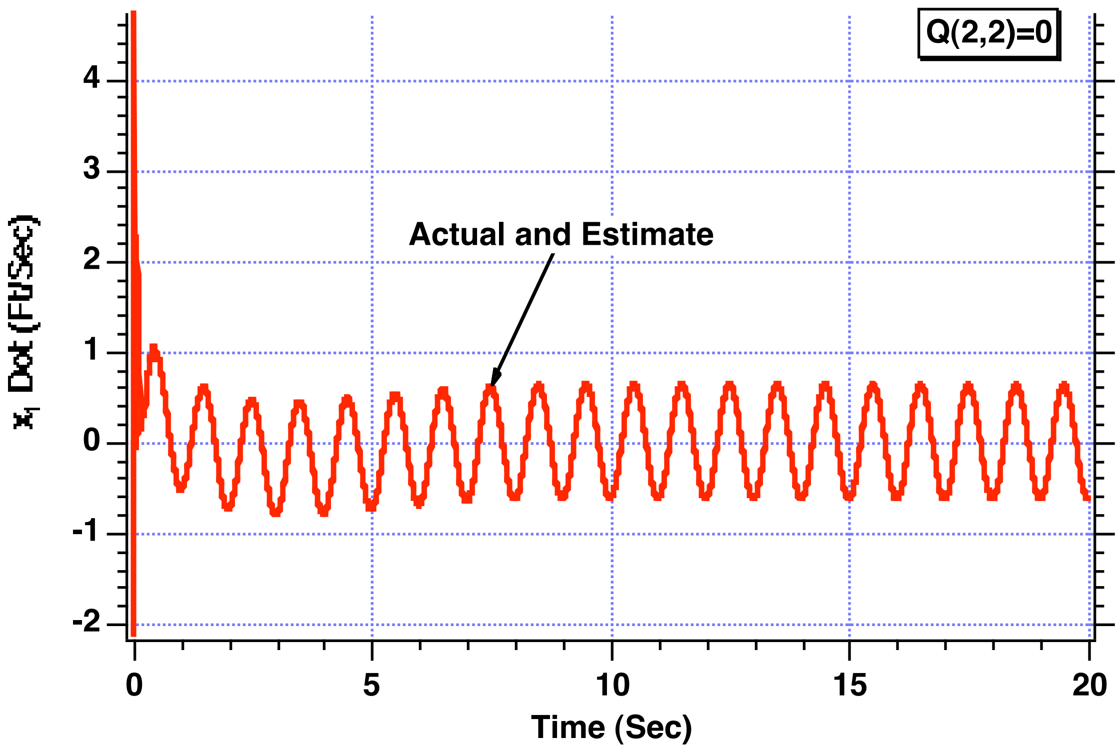
Write data to screen and files

Subroutine to generate Gaussian noise

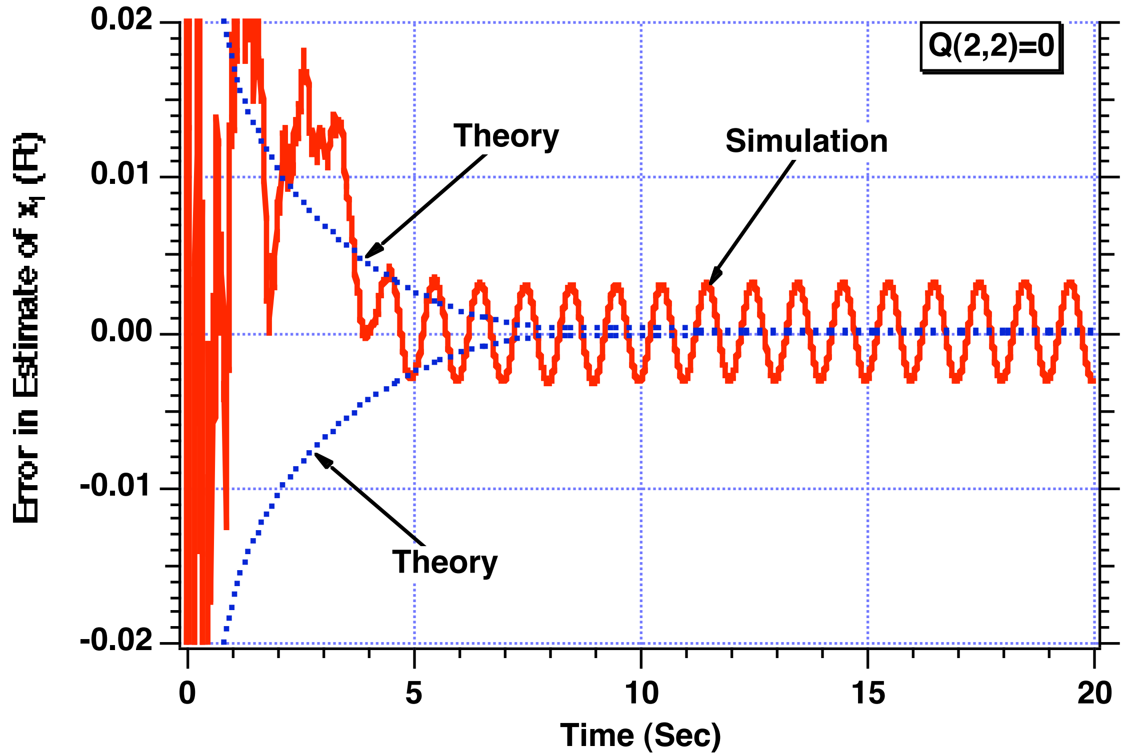
Kalman Filter Provides Excellent Estimates of the First State of the Suspension



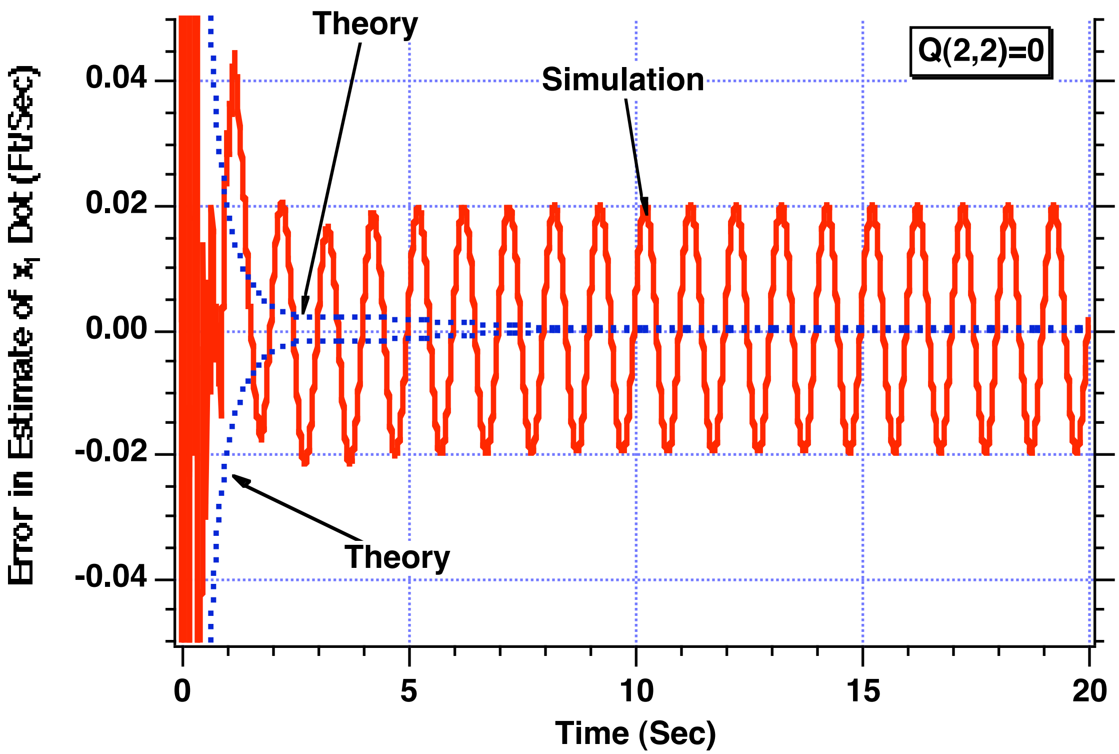
Kalman Filter Provides Excellent Estimates of the Derivative of the First State of the Suspension



Error in the Estimate of First State Does Not Agree With Covariance Matrix Predictions



Error in the Estimate of Second State Does Not Agree With Covariance Matrix Predictions

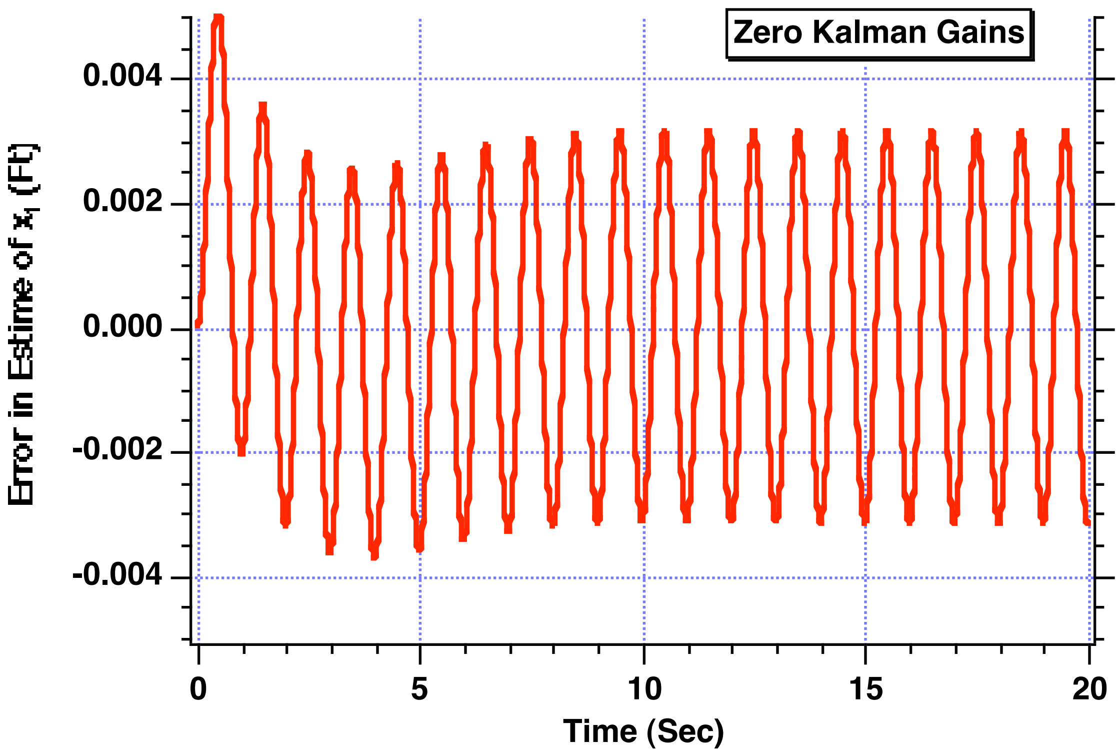


Testing For What Went Wrong

Coast filter by setting Kalman gains to zero and perfectly initializing filter

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

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

Kalman Filter Does Not Coast Properly When Filter is Perfectly Initialized

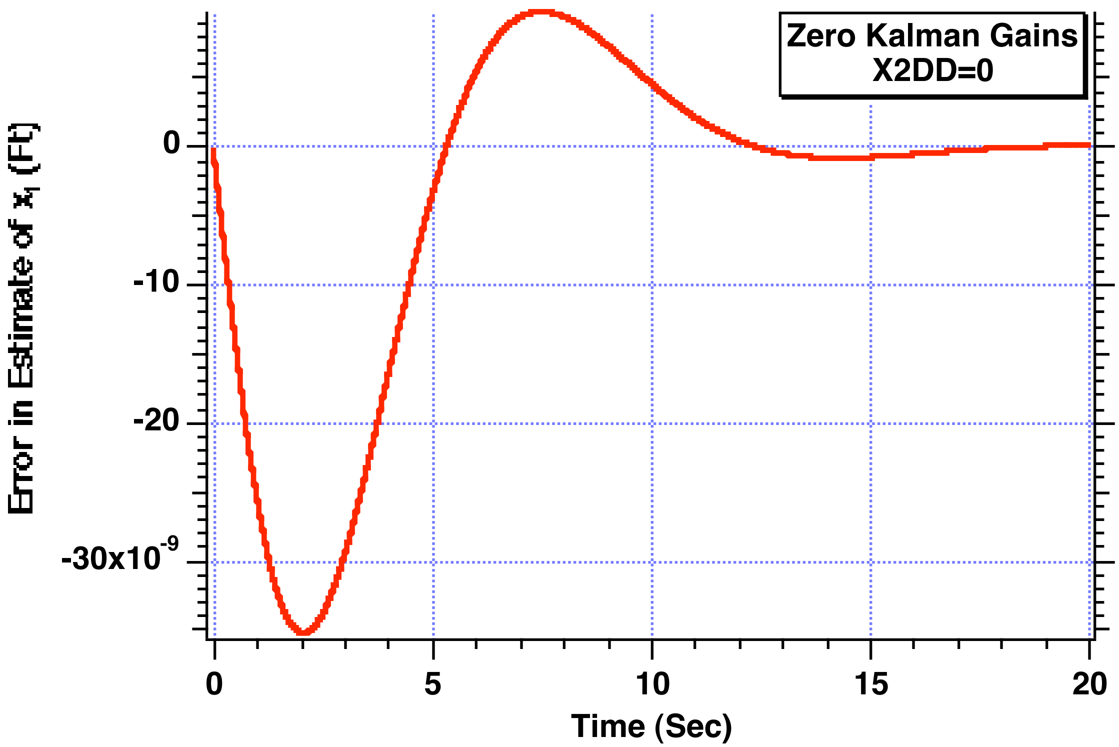


Conclusion: Either Φ or G matrix is wrong because error in estimate does not go to zero

Simulation Further Simplified By Setting X2DD to Zero

This means we focus on fundamental matrix only

Fundamental Matrix Appears to be Correct

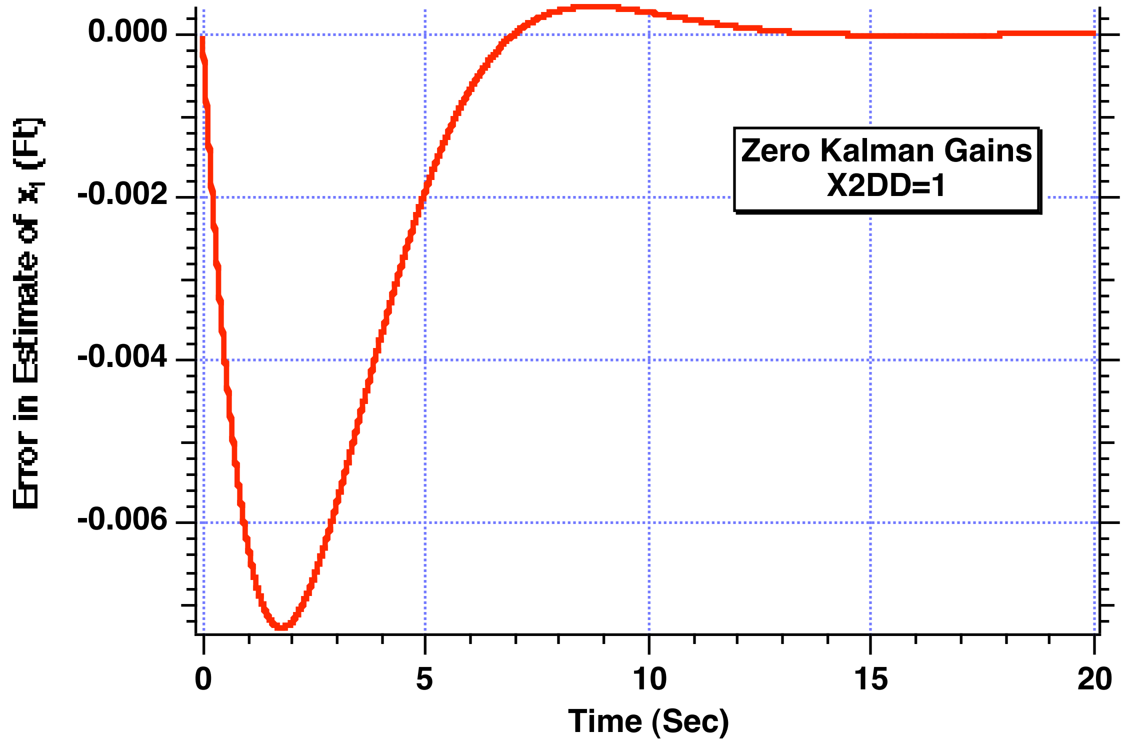


Conclusion: G matrix in error

Strictly Speaking G Correct When X2DD Constant Between Sampling Instants

Let us set X2DD=1 for test

Discrete G Matrix is Correct if Deterministic Input is Constant

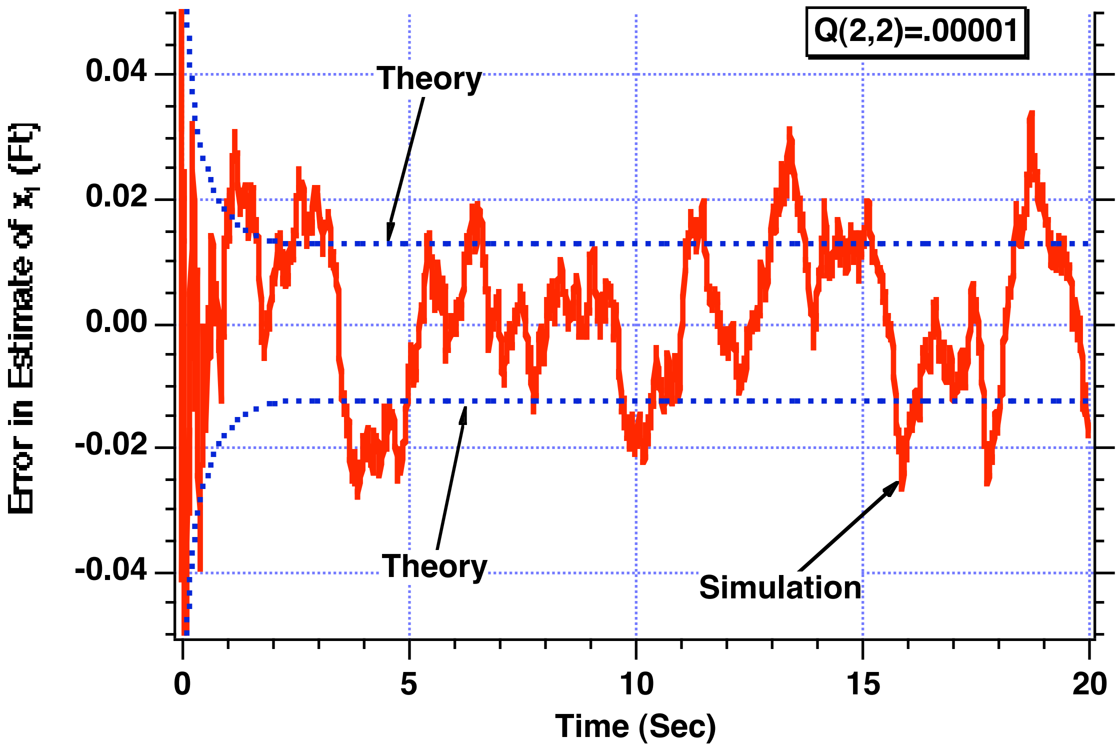


Conclusion: G matrix not correct because X2DD not constant Between sampling instants

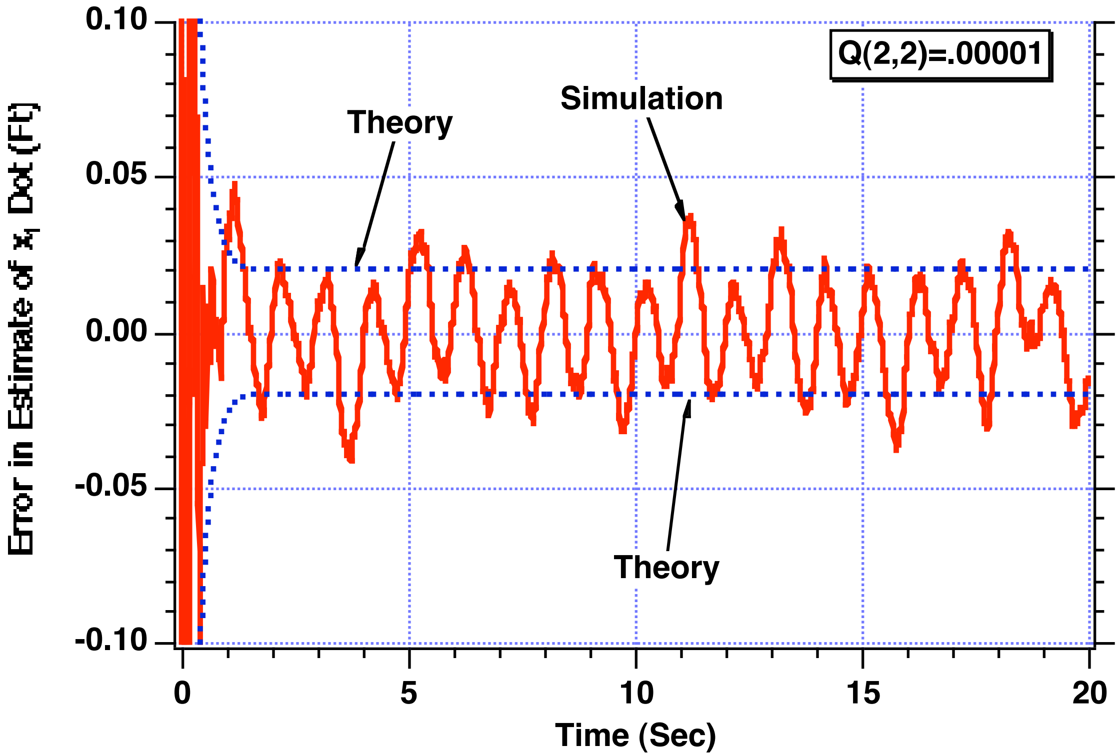
Two Ways to Go

- Rederive G
  - Lots of math
- Add process noise
  - Easy to try

Addition of Process noise ensures that Errors in the Estimate of the First State are Within the Theoretical Error Bounds



Addition of Process noise ensures that Errors in the Estimate of the Second State are Within the Theoretical Error Bounds



Kalman Filters in a Non Polynomial World
Summary

- Polynomial Kalman filters may not give good results when measurement signal is not a polynomial
- It is best to have accurate fundamental matrix
    - Accuracy of fundamental matrix is more important in the filter than in the Riccati equations
    - Adding process noise can help with inaccuracies
- Two non polynomial filters designed