Miscellaneous Adaptive Topics

Miscellaneous Adaptive Topics Overview

- Filters in Parallel
- Adjusting Process Noise

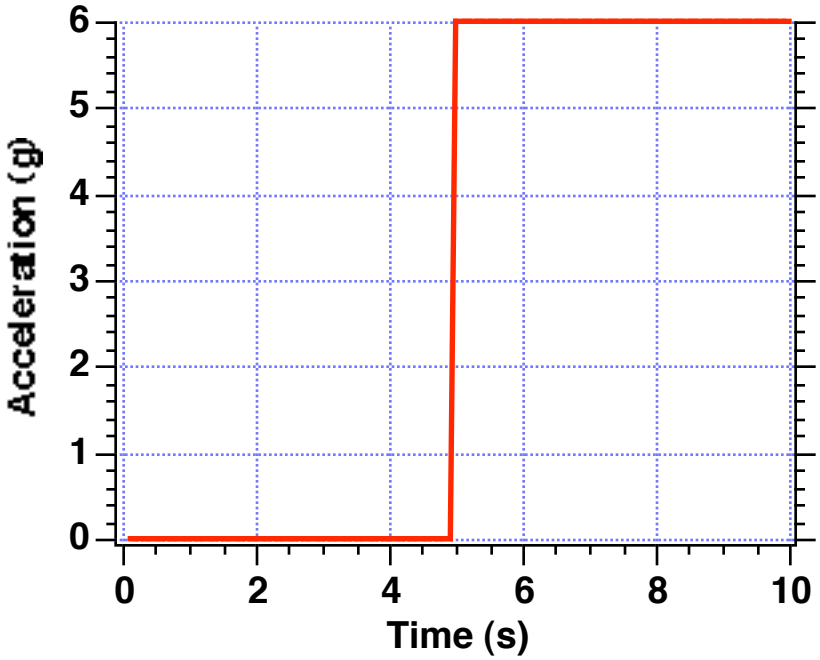
Making Filters Adaptive

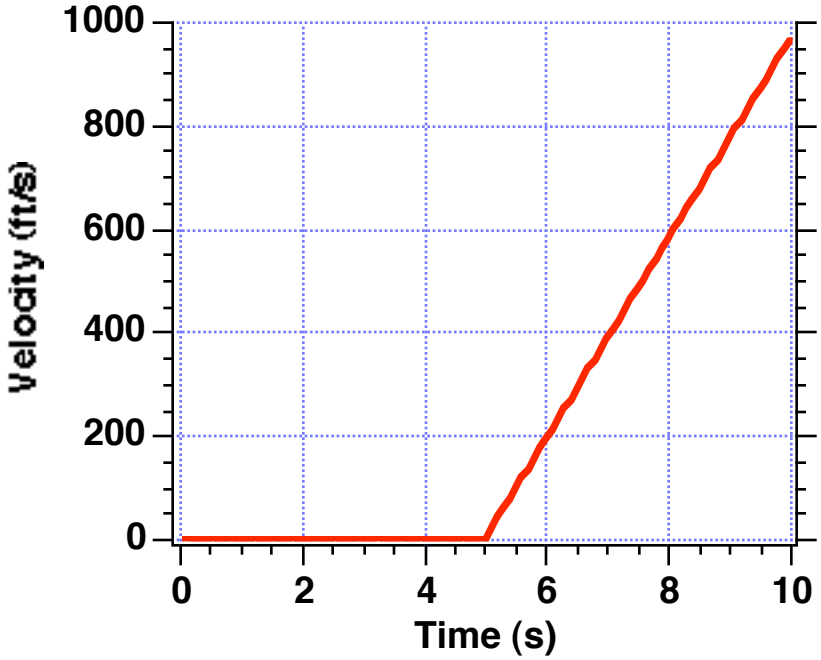
Making Filters Adaptive
Overview

- Tracking a maneuvering and non maneuvering target
  - Problems with individual two-state and three-state filters
  - Combining filters for best performance
- Tracking a boosting target that eventually goes ballistic
  - Problems with individual two-state and three-state filters
  - Combining filters for best performance

Tracking a Maneuvering and Non Maneuvering Target

Target Model





Important Matrices for Different Order Polynomial Kalman Filters

Order	Systmes Dynamics	Fundamental	Measurement	Noise
0	$\mathbf{F}=1$	$\Phi_k=1$	$\mathbf{H}=1$	$\mathbf{R}_k=\sigma_n^2$
1	$\mathbf{F}=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}$	$\Phi_k=\begin{bmatrix}1 & T_s\\ 0 & 1\end{bmatrix}$	$\mathbf{H}=[1 \quad 0]$	$\mathbf{R}_k=\sigma_n^2$
2	$\mathbf{F}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}$	$\Phi_k=\begin{bmatrix}1 & T_s & .5T_s^2\\ 0 & 1 & T_s\\ 0 & 0 & 1\end{bmatrix}$	$\mathbf{H}=[1 \quad 0 \quad 0]$	$\mathbf{R}_k=\sigma_n^2$

Two-State Polynomial Kalman Filter

Kalman filtering equation

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

Substitution of matrices yields

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

Multiplying out the matrices

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

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

If we define the residual to be

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

The filter becomes

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

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

Three-State Polynomial Kalman Filter

Kalman filtering equation

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

Substitution of matrices yields

$$\begin{bmatrix} \hat{x}_k \\ \hat{\dot{x}}_k \\ \hat{\ddot{x}}_k \end{bmatrix} = \begin{bmatrix} 1 & T_s & .5T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \\ \hat{\ddot{x}}_{k-1} \end{bmatrix} + \begin{bmatrix} K_{1_k} \\ K_{2_k} \\ K_{3_k} \end{bmatrix} \left[x_k^* - \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & T_s & .5T_s^2 \\ 0 & 1 & T_s \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \\ \hat{\ddot{x}}_{k-1} \end{bmatrix} \right]$$

Multiplying out the matrices

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

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

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

If we define the residual to be

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

The filter becomes

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

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

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

The Discrete Process Noise Matrix Varies With System Order

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

Maneuvering and Non Maneuvering Target Code-1

```
GLOBAL DEFINE
                INCLUDE 'quickdraw.inc'
END
IMPLICIT REAL*8(A-H,O-Z)
REAL*8 P(3,3),Q(3,3),M(3,3),PHI(3,3),HMAT(1,3),HT(3,1),PHIT(3,3)
REAL*8 RMAT(1,1),IDN(3,3),PHIP(3,3),PHIPPHIT(3,3),HM(1,3)
REAL*8 HMHT(1,1),HMHTR(1,1),HMHTRINV(1,1),MHT(3,1),K(3,1)
REAL*8 KH(23,3),IKH(3,3)
REAL*8 PZ(2,2),QZ(2,2),MZ(2,2),PHIZ(2,2),HMATZ(1,2),HTZ(2,1)
REAL*8 PHITZ(2,2)
REAL*8 RMATZ(1,1),IDNZ(2,2),PHIPZ(2,2),PHIPPHITZ(2,2),HMZ(1,2)
REAL*8 HMHTZ(1,1),HMHTRZ(1,1),HMHTRINVZ(1,1),MHTZ(2,1),KZ(2,1)
REAL*8 KHZ(2,2),IKHZ(2,2)
INTEGER ORDER,STEP
INTEGER ORDERZ
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
XNT=193.2
YT=0.
YTD=0.
ORDER=3
ORDERZ=2
PHIS=XNT*XNT/10.
PHISZ=10.*10./10.
TS=.1
XH=0.
XDH=0.
XDDH=0
ZH=0.
ZDH=0.
SIGNOISE=50.
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.
```

Initial Conditions

Process Noise For 3SKF and 2SKF

Initial State Estimates For 2SKF and 3SKF

Maneuvering and Non Maneuvering Target Code-2

```
14      CONTINUE
        DO 144 I=1,ORDERZ
        DO 144 J=1,ORDERZ
        PHIZ(I,J)=0.
        PZ(I,J)=0.
        QZ(I,J)=0.
        IDNZ(I,J)=0.
144     CONTINUE
        RMAT(1,1)=SIGNOISE**2
        RMATZ(1,1)=SIGNOISE**2
        IDN(1,1)=1.
        IDN(2,2)=1.
        IDN(3,3)=1.
        IDNZ(1,1)=1.
        IDNZ(2,2)=1.
        P(1,1)=99999999999.
        P(2,2)=99999999999.
        P(3,3)=99999999999.
        PZ(1,1)=99999999999.
        PZ(2,2)=99999999999.
        PHI(1,1)=1
        PHI(1,2)=TS
        PHI(1,3)=.5*TS*TS
        PHI(2,2)=1
        PHI(2,3)=TS
        PHI(3,3)=1
        PHIZ(1,1)=1
        PHIZ(1,2)=TS
        PHIZ(2,2)=1
        HMAT(1,1)=1.
        HMAT(1,2)=0.
        HMAT(1,3)=0.
        HMATZ(1,1)=1.
        HMATZ(1,2)=0.
        CALL MATTRN(PHI,ORDER,ORDER,PHIT)
        CALL MATTRN(HMAT,1,ORDER,HT)
        CALL MATTRN(PHIZ,ORDERZ,ORDERZ,PHITZ)
        CALL MATTRN(HMATZ,1,ORDERZ,HTZ)
```

Initial Covariance Matrices For 3SKF and 2SKF

Fundamental Matrices For 3SKF and 2SKF

Measurement Matrices For 3SKF and 2SKF

Transposes

Maneuvering and Non Maneuvering Target Code-3

```
      Q(1,1)=PHIS*TS**5/20
      Q(1,2)=PHIS*TS**4/8
      Q(1,3)=PHIS*TS**3/6
      Q(2,1)=Q(1,2)
      Q(2,2)=PHIS*TS**3/3
      Q(2,3)=PHIS*TS*TS/2
      Q(3,1)=Q(1,3)
      Q(3,2)=Q(2,3)
      Q(3,3)=PHIS*TS
      QZ(1,1)=PHISZ*TS**3/3
      QZ(1,2)=PHISZ*TS*TS/2
      QZ(2,1)=QZ(1,2)
      QZ(2,2)=PHISZ*TS
      S=0.
      H=.001
      T=0.
      XN=1.
10    IF(T>10.)GOTO 999
      S=S+H
      YTOLD=YT
      YTDOLD=YTD
      STEP=1
      GOTO 200
66    STEP=2
      YT=YT+H*YTD
      YTD=YTD+H*YTDD
      T=T+H
      GOTO 200
55    CONTINUE
      YT=.5*(YTOLD+YT+H*YTD)
      YTD=.5*(YTDOLD+YTD+H*YTDD)
```

Process Noise Matrices For 3SKF and 2SKF

Second-Order Runge Kutta Integration

Maneuvering and Non Maneuvering Target Code-4

```
      IF(S<(TS-.00001))GOTO 10
      S=0.
      CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
      CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,PHIPPHIT)
      CALL MATADD(PHIPPHIT,ORDER,ORDER,Q,M)
      CALL MATMUL(HMAT,1,ORDER,M,ORDER,ORDER,HM)
      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 MATMUL(PHIZ,ORDERZ,ORDERZ,PZ,ORDERZ,ORDERZ,PHIPZ)
      CALL MATMUL(PHIPZ,ORDERZ,ORDERZ,PHITZ,ORDERZ,ORDERZ,
     1                  PHIPPHITZ)
      CALL MATADD(PHIPPHITZ,ORDERZ,ORDERZ,QZ,MZ)
      CALL MATMUL(HMATZ,1,ORDERZ,MZ,ORDERZ,ORDERZ,HMZ)
      CALL MATMUL(HMZ,1,ORDERZ,HTZ,ORDERZ,1,HMHTZ)
      CALL MATADD(HMHTZ,ORDERZ,ORDERZ,RMATZ,HMHTRZ)
      HMHTRINVZ(1,1)=1./HMHTRZ(1,1)
      CALL MATMUL(MZ,ORDERZ,ORDERZ,HTZ,ORDERZ,1,MHTZ)
      CALL MATMUL(MHTZ,ORDERZ,1,HMHTRINVZ,1,1,KZ)
      CALL MATMUL(KZ,ORDERZ,1,HMATZ,1,ORDERZ,KHZ)
      CALL MATSUB(IDNZ,ORDERZ,ORDERZ,KHZ,IKHZ)
      CALL MATMUL(IKHZ,ORDERZ,ORDERZ,MZ,ORDERZ,ORDERZ,PZ)
      CALL GAUSS(XNOISE,SIGNOISE)
      XK1LS=2.*(2.*XN-1.)/(XN*(XN+1.))
      XK2LS=6./(XN*(XN+1.))
      XK1PLS=3.*(3.*XN*XN-3.*XN+2.)/(XN*(XN+1.)*(XN+2.))
      XK2PLS=18.*(2.*XN-1.)/(XN*(XN+1.)*(XN+2.))
      XK3PLS=60./(XN*(XN+1.)*(XN+2.))
      XN=XN+1.
      X=YT
      XD=YTD
      XDD=YTDD
      XS=X+XNOISE
```

Riccati Equations For 3SKF

Riccati Equations For 2SKF

Gains For 2-State and 3-State Least Squares Filters

Maneuvering and Non Maneuvering Target Code-5

```
IF(XK1PLS>K(1,1))THEN
                XK1=XK1PLS
                XK2=XK2PLS
                XK3=XK3PLS
ELSE
                XK1=K(1,1)
                XK2=K(2,1)
                XK3=K(3,1)
ENDIF
RES=XS-XH-TS*XDH-.5*TS*TS*XDDH
XH=XH+XDH*TS+.5*TS*TS*XDDH+XK1*RES
XDH=XDH+XDDH*TS+XK2*RES
XDDH=XDDH+XK3*RES
```

3SKF

```
IF(XK1LS>KZ(1,1))THEN
                XK1P=XK1LS
                XK2P=XK2LS
ELSE
                XK1P=KZ(1,1)
                XK2P=KZ(2,1)
ENDIF
RESZ=XS-ZH-TS*ZDH
ZH=ZH+ZDH*TS+XK1P*RESZ
ZDH=ZDH+XK2P*RESZ
```

2SKF

```
ERRX=X-XH
ERRZ=X-ZH
ERRXD=XD-XDH
ERRZD=XD-ZDH
SRES=SQRT(HMHTRZ(1,1))
IF(ABS(RESZ)>2.*SRES)THEN
                XHPZ=XH
                XDHPZ=XDH
                IFILTER=3
ELSE
                XHPZ=ZH
                XDHPZ=ZDH
                IFILTER=2
ENDIF
```

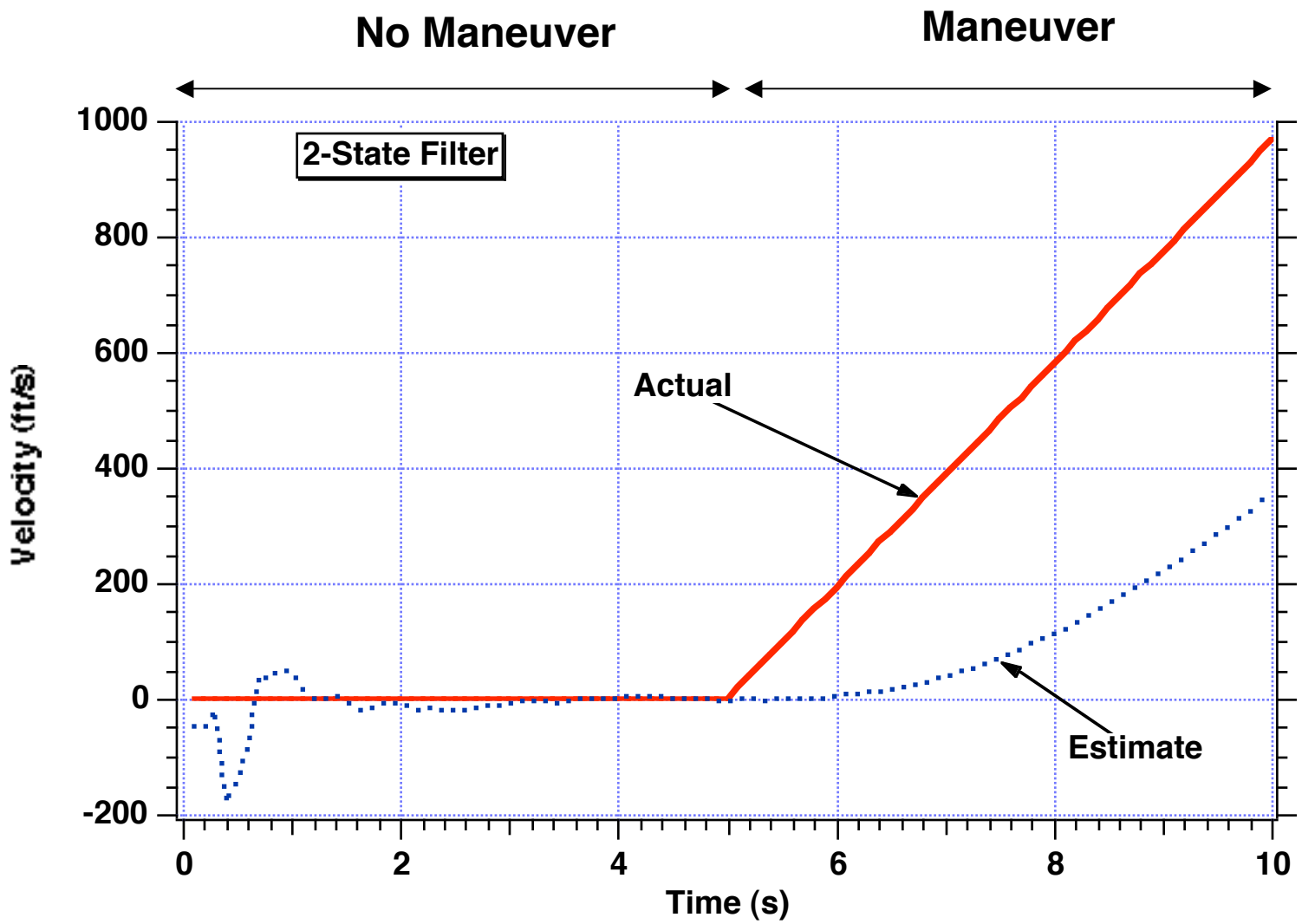
Logic For Choosing Filter

Maneuvering and Non Maneuvering Target Code-6

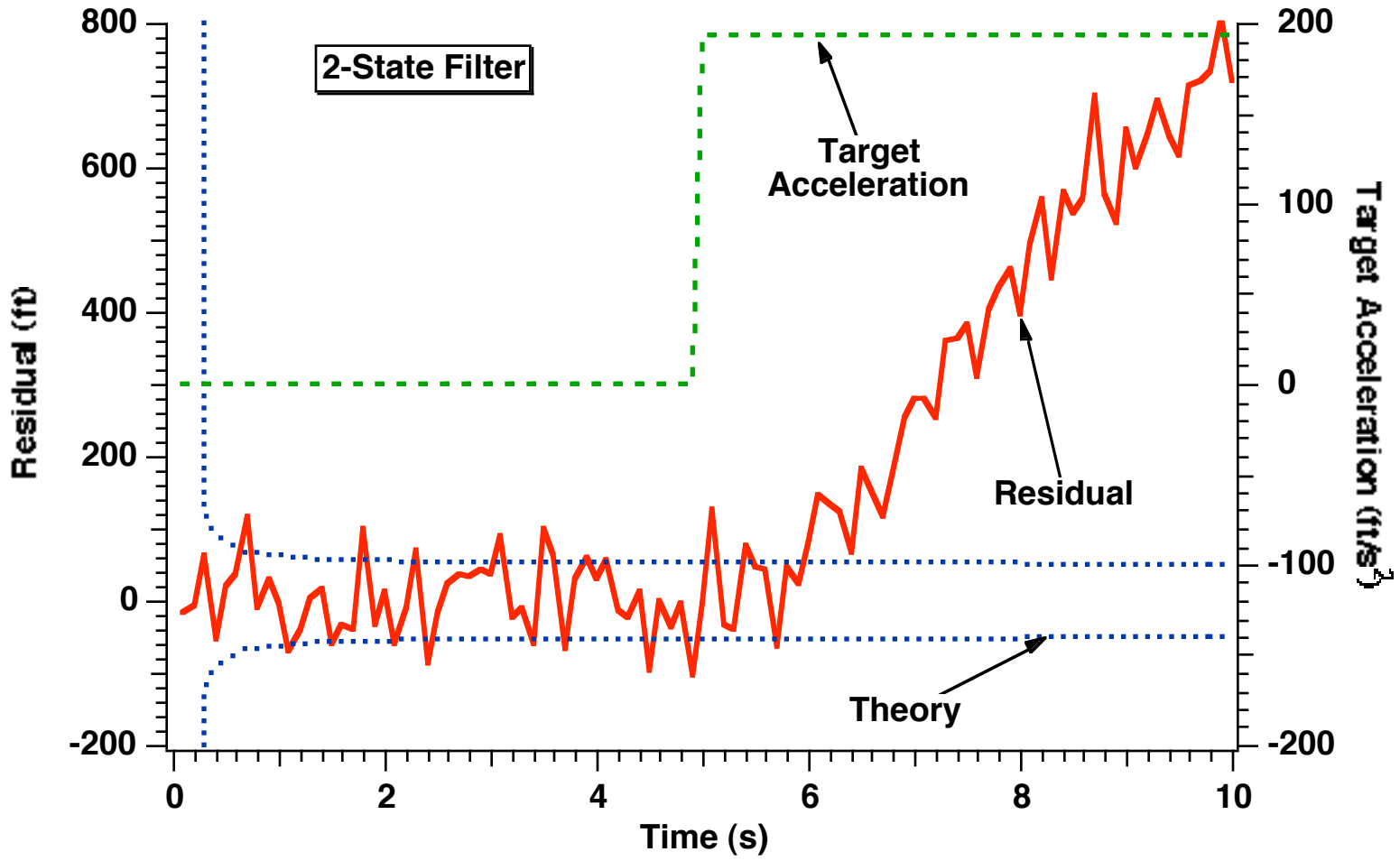
```
      WRITE(9,*)T,X,XH,ZH,XHPZ,XD,XDH,ZDH,XDHPZ,XDD,XDDH,
     1                                    RES,RESZ,IFILTER
      WRITE(1,*)T,X,XH,ZH,XHPZ,XD,XDH,ZDH,XDHPZ,XDD,XDDH,
     1                                    RES,RESZ,IFILTER
      WRITE(2,*)T,ERRX,ERRZ,ERRXD,ERRZD
      GOTO 10
200   CONTINUE
      IF(T<5.)THEN
                  YTDD=0.
      ELSE
                  YTDD=XNT
      ENDIF
      IF(STEP-1)66,66,55
999   CONTINUE
      PAUSE
      CLOSE(1)
      CLOSE(2)
      END
```

Differential Equation Representing Real World

Two-State Kalman Filter Diverges When Target Maneuvers

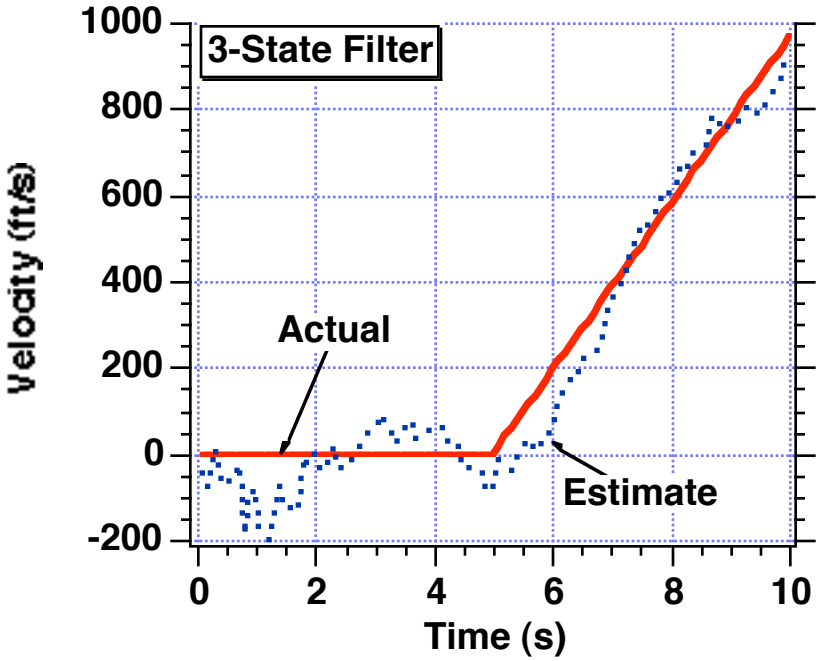


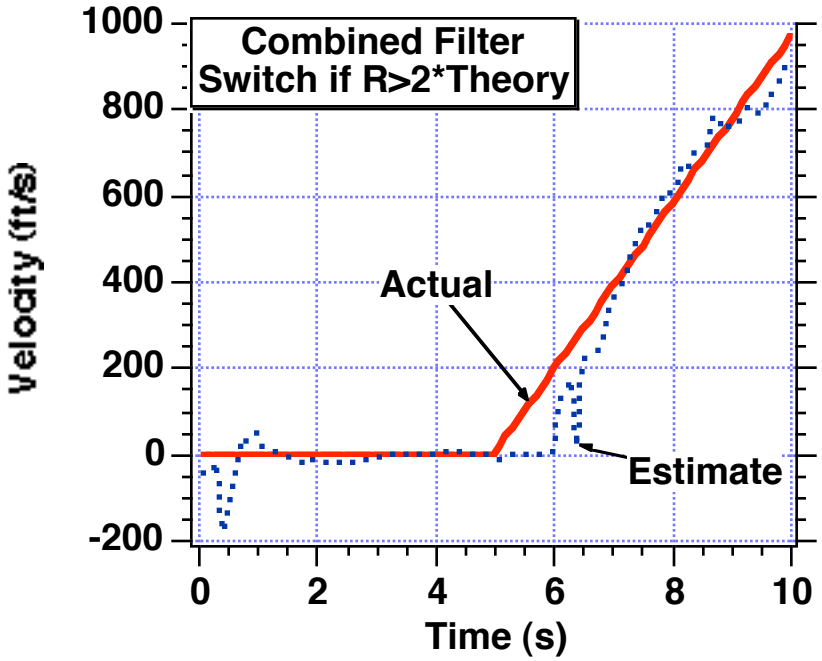
Residual of Two-State Kalman Filter Increases When Target Maneuvers



$$E(RES * RES^T) = H_k M_k H^T + R_k$$

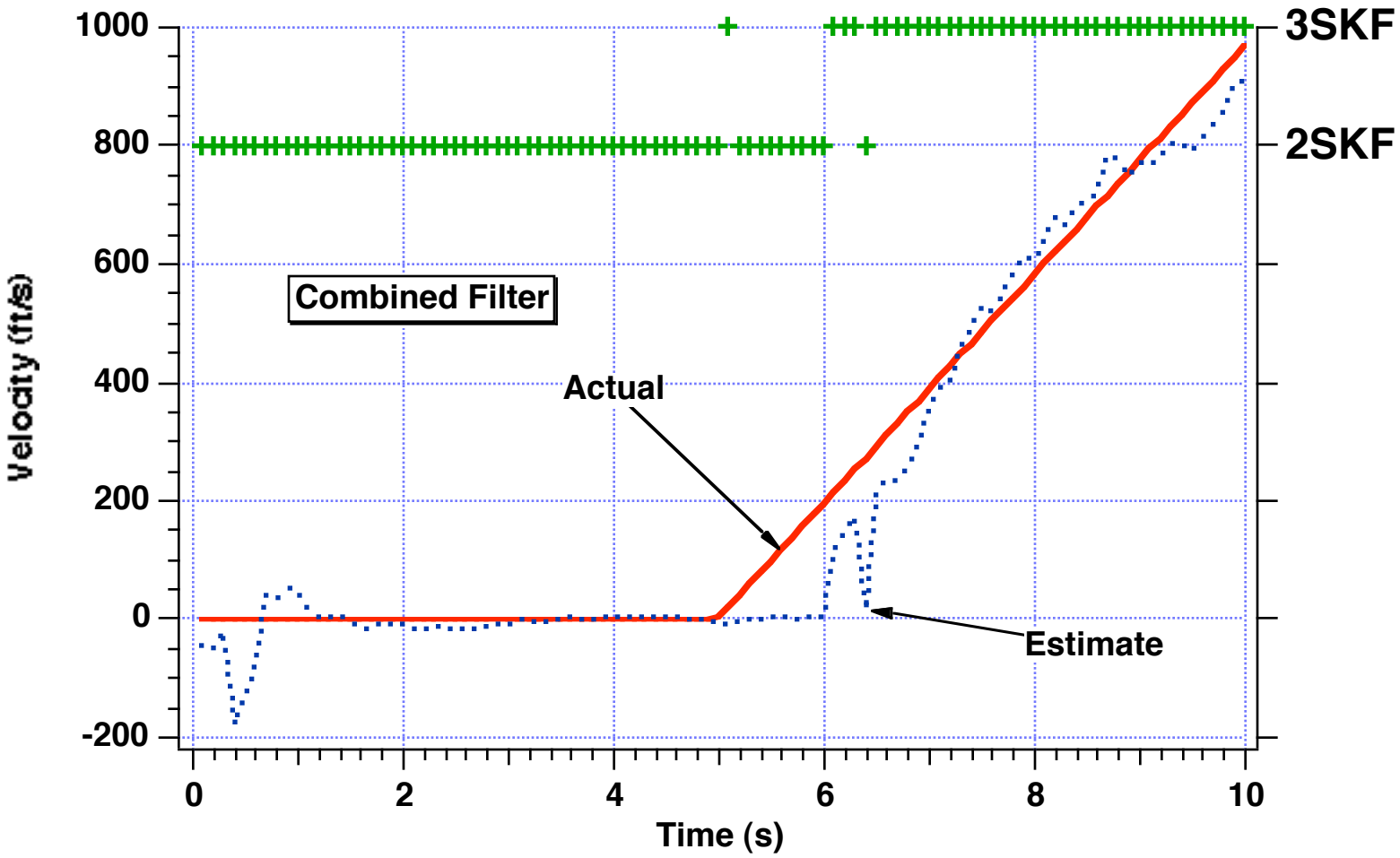
Combined Filter Provides Better Estimates



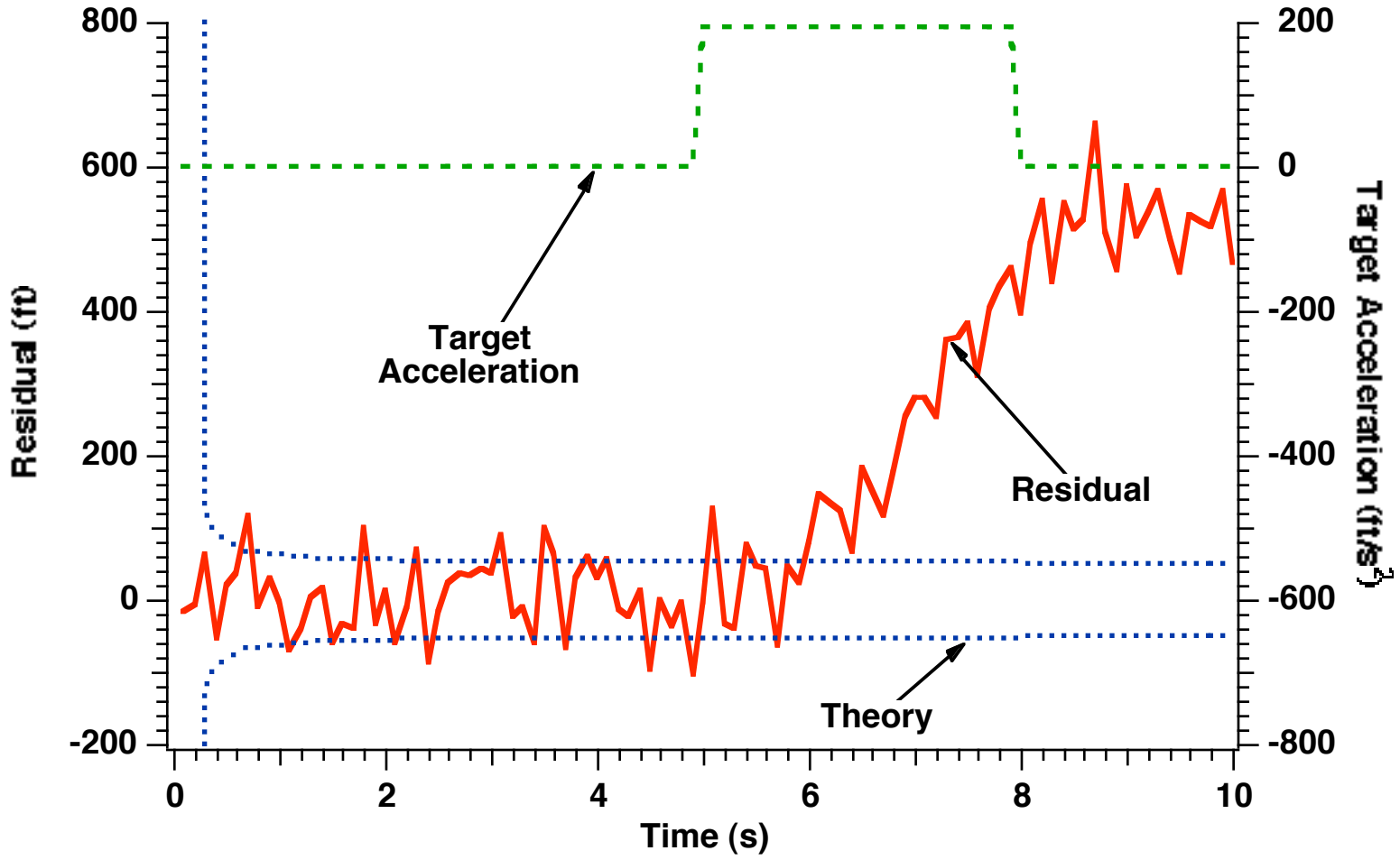


$$E(RES * RES^T) = H_k M_k H^T + R_k$$

Adaptive Scheme Works Near Perfectly

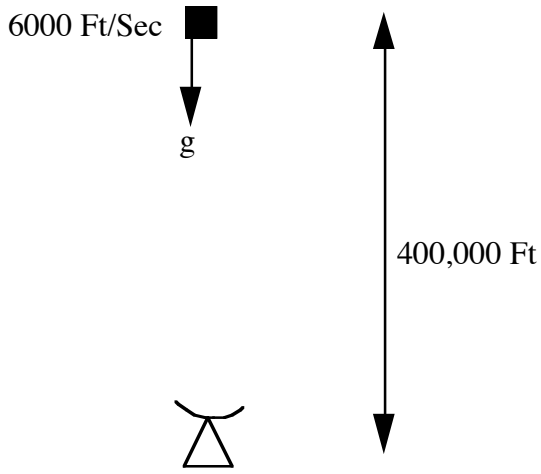


Residual Logic As Shown Is Not Bullet Proof



Making The Process Noise Adaptive

Radar Tracking Falling Object



From basic physics

$$x = 400000 - 6000t - \frac{gt^2}{2}$$ ← **Second-order process**

Velocity of object can be found by differentiating

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

Radar measures altitude with standard deviation of 1000 ft

Desire to track object and estimate altitude and velocity

First-Order Polynomial Kalman Filter (No Gravity Compensation)

Kalman filtering equation

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

Substitution of matrices yields

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

Multiplying out the matrices

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

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

If we define the residual to be

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

The filter becomes

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

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

Important Matrices for Different Order Polynomial Kalman Filters

Order	Systmes Dynamics	Fundamental	Measurement	Noise
0	$\mathbf{F}=1$	$\Phi_k=1$	$\mathbf{H}=1$	$\mathbf{R}_k=\sigma_n^2$
1	$\mathbf{F}=\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}$	$\Phi_k=\begin{bmatrix}1 & T_s\\ 0 & 1\end{bmatrix}$	$\mathbf{H}=[1 \quad 0]$	$\mathbf{R}_k=\sigma_n^2$
2	$\mathbf{F}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{bmatrix}$	$\Phi_k=\begin{bmatrix}1 & T_s & .5T_s^2\\ 0 & 1 & T_s\\ 0 & 0 & 1\end{bmatrix}$	$\mathbf{H}=[1 \quad 0 \quad 0]$	$\mathbf{R}_k=\sigma_n^2$

The Discrete Process Noise Matrix Varies With System Order

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

Filter gains are obtained from Riccati equations

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

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

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

MATLAB Simulation of First-Order Polynomial Kalman Filter and Falling Object-1

```
ORDER =2;
PHIS=0.;
TS=.1;
A0=400000;
A1=-6000.;
A2=-16.1;
XH=0;
XDH=0;
SIGNOISE=1000.;
PHI=[1 TS;0 1];
P=[99999999 0;0 999999999];
IDNP=eye(ORDER);
Q=zeros(ORDER);
H=[1 0];
HT=H';
R=SIGNOISE^2;
PHIT=PHI';
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;
count=0;
for T=0:TS:30
        PHIP=PHI*P;
        PHIPPHIT=PHIP*PHIT;
        M=PHIPPHIT+Q;
        MHT=M*HT;
        HMHT=H*MHT;
        HMHTR=HMHT+R;
        HMHTRINV=inv(HMHTR);
        K=MHT*HMHTRINV;
        KH=K*H;
        IKH=IDNP-KH;
        P=IKH*M;
```

System order

Process noise

Fundamental and initial covariance matrices

Measurement matrix

Process noise matrix

Riccati equations

MATLAB Simulation of First-Order Polynomial Kalman Filter and Falling Object -2

```
XNOISE=SIGNOISE*randn;
X=A0+A1*T+A2*T*T;
XD=A1+2*A2*T;
XS=X+XNOISE;
RES=XS-XH-TS*XDH;
XH=XH+XDH*TS+K(1,1)*RES;
XDH=XDH+K(2,1)*RES;
SP11=sqrt(P(1,1));
SP22=sqrt(P(2,2));
XHERR=X-XH;
XDHERR=XD-XDH;
SP11P=-SP11;
SP22P=-SP22;
count=count+1;
ArrayT(count)=T;
ArrayX(count)=X;
ArrayXH(count)=XH;
ArrayXD(count)=XD;
ArrayXDH(count)=XDH;
ArrayXHERR(count)=XHERR;
ArraySP11(count)=SP11;
ArraySP11P(count)=SP11P;
ArrayXDHERR(count)=XDHERR;
ArraySP22(count)=SP22;
ArraySP22P(count)=SP22P;
end
```

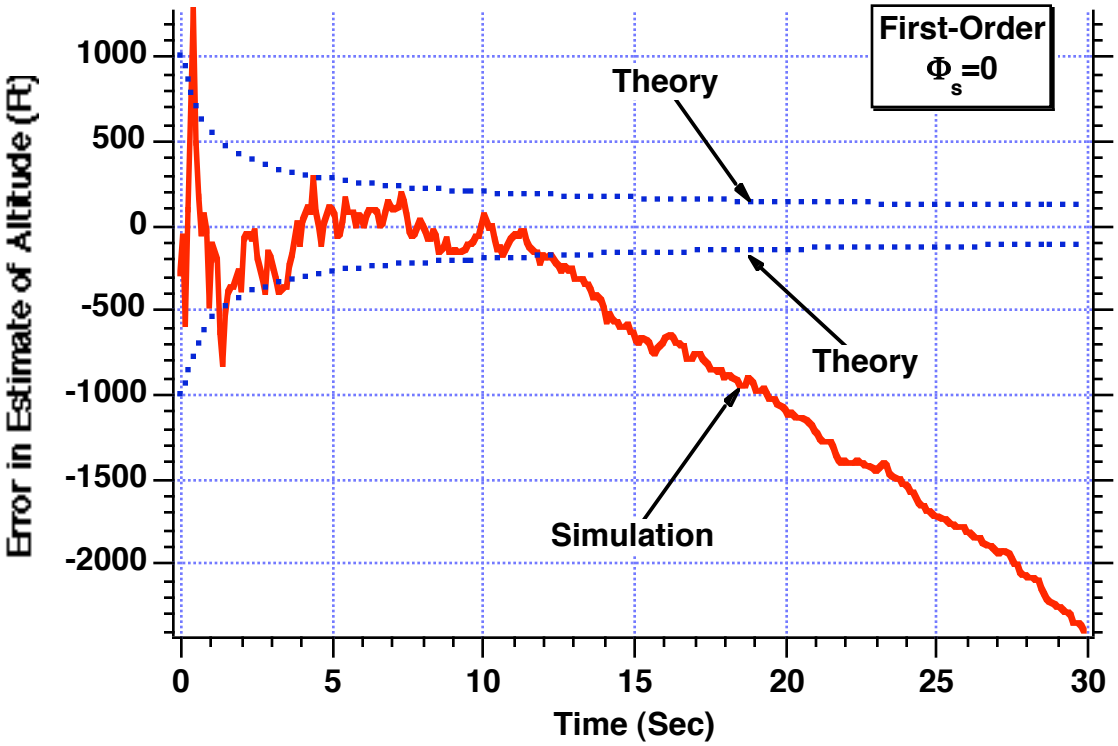
Signal

Kalman filter

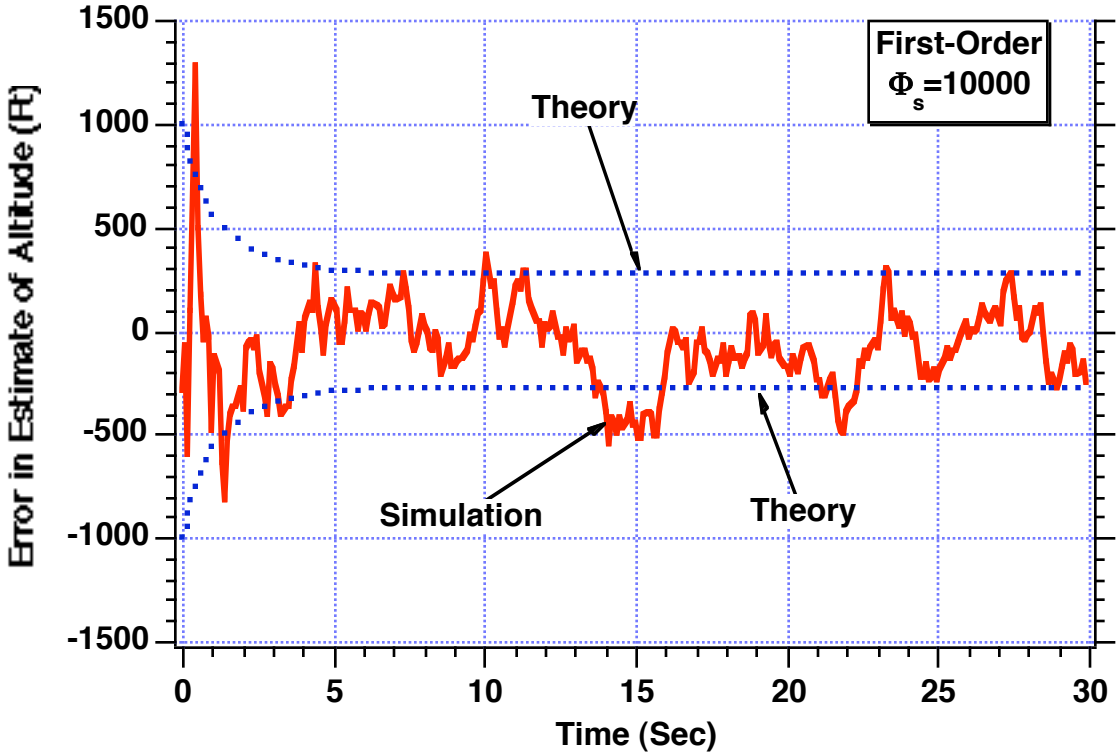
Errors in estimates

Save data for plotting and writing to files

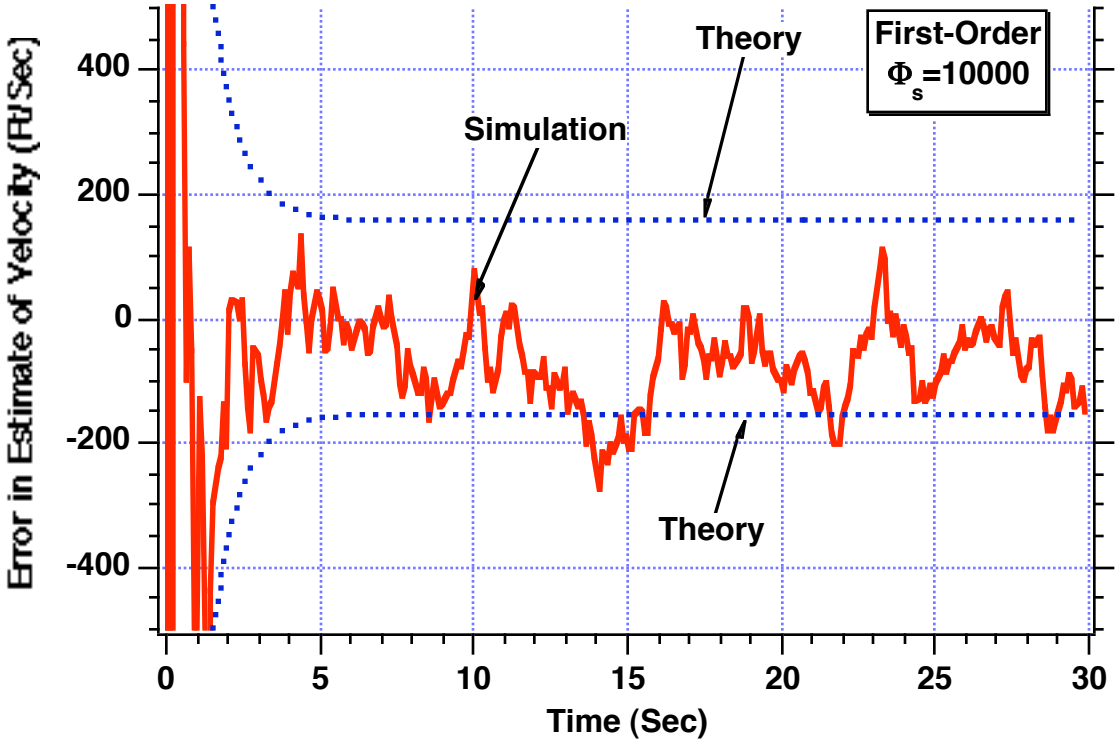
First-Order Polynomial Kalman Filter Without Process Noise Can Not Track Second-Order Signal



Adding Process Noise Prevents Altitude Errors of First-Order Filter From Diverging



Adding Process Noise Prevents Velocity Errors of First-Order Filter From Diverging



Code For Adaptive Process Noise - 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)
      REAL*8 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')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      T=0.
      TF=30.
      A0=400000.
      A1=-6000.
      A2=-16.1
      X=A0+A1*T+A2*T*T
      XD=A1+2.*A2*T
      ORDER=2
      PHIS=0.
      TS=.1
      XH=X
      XDH=XD
      SIGNOISE=1000.
      DO 144 I=1,ORDER
      DO 144 J=1,ORDER
      PHI(I,J)=0.
      P(I,J)=0.
      Q(I,J)=0.
      IDN(I,J)=0.
144   CONTINUE
      RMAT(1,1)=SIGNOISE**2
      IDN(1,1)=1.
      IDN(2,2)=1.
      P(1,1)=99999999999.
      P(2,2)=99999999999.
```

Code For Adaptive Process Noise - 2

```
      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)
      DO 10 T=0.,TF,TS
                  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
                  CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
                  CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,
     1                        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)
```

Code For Adaptive Process Noise - 3

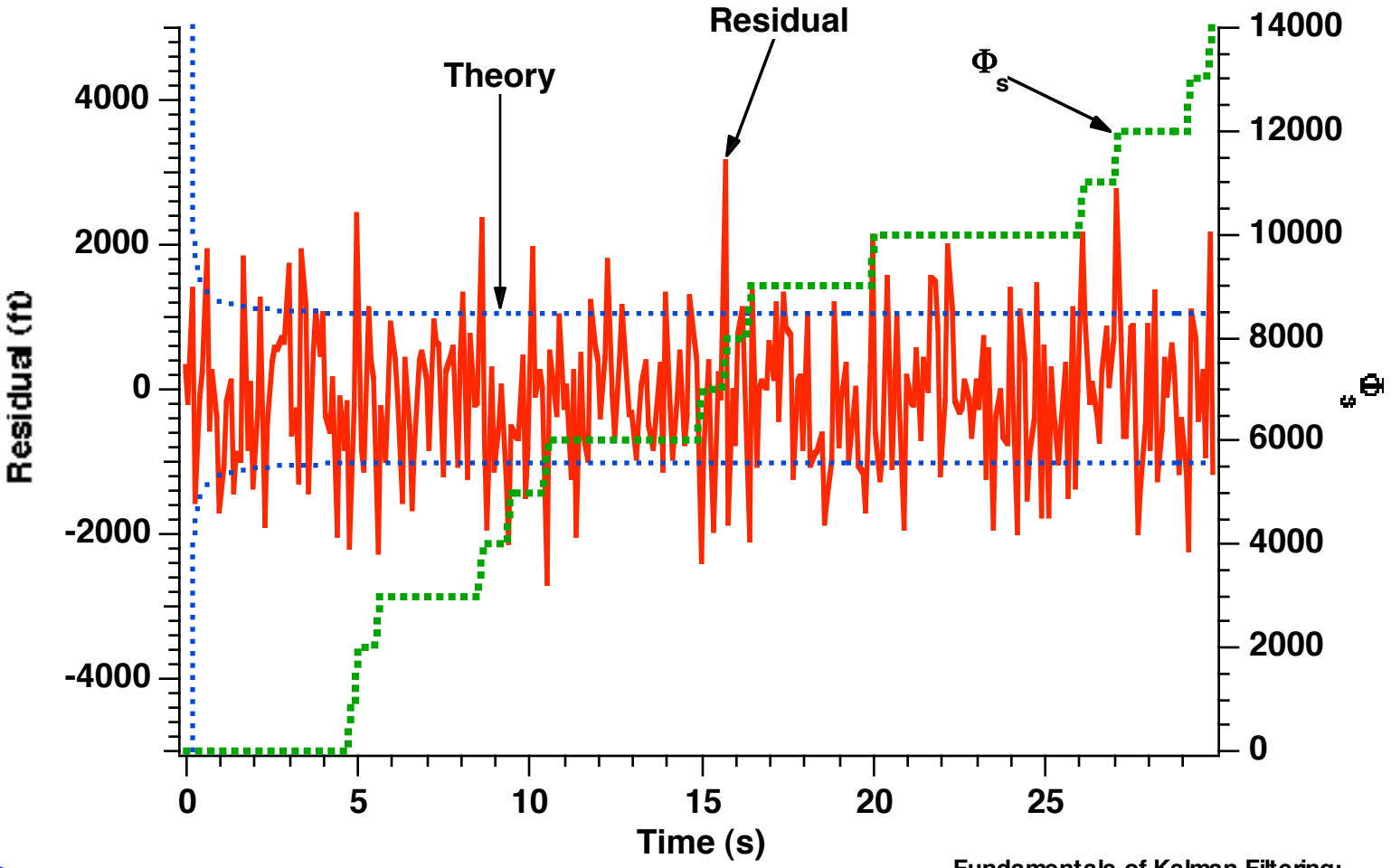
```
                X=A0+A1*T+A2*T*T
                XD=A1+2.*A2*T
                CALL GAUSS(XNOISE,SIGNOISE)
                XS=X+XNOISE
                RES=XS-XH-TS*XDH
                XH=XH+XDH*TS+K(1,1)*RES
                XDH=XDH+K(2,1)*RES
                ERRX=X-XH
                ERRXD=XD-XDH
                SP11=SQRT(P(1,1))
                SP22=SQRT(P(2,2))
                SPRES=SQRT(HMHTR(1,1))
                IF(ABS(RES)>2.*SPRES)THEN
                        PHIS=PHIS+1000.
                ENDIF
                WRITE(9,*)T,X,XH,XD,XDH,PHIS
                WRITE(1,*)T,X,XH,XD,XDH,PHIS
                WRITE(2,*)T,ERRX,SP11,-SP11,ERRXD,SP22,-SP22,RES,
     1          SPRES,-SPRES
 10     CONTINUE
        PAUSE
        CLOSE(1)
        CLOSE(2)
        END
```

Residual is Used To Increase Process Noise Matrix

$$\Phi_{s_0} = 0$$

If (| Residual | > 2*Theory) Then

$$\Phi_{s_k} = \Phi_{s_{k-1}} + 1000$$

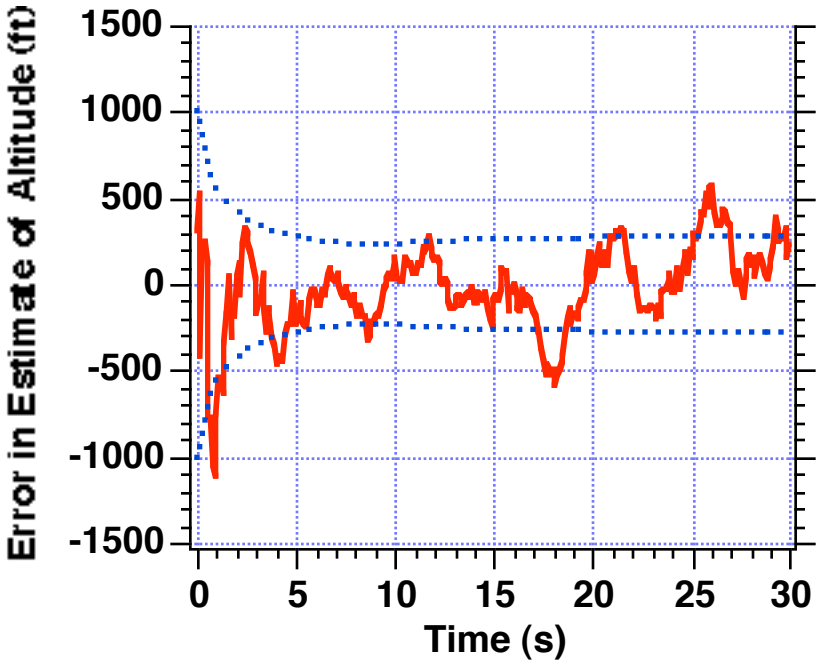


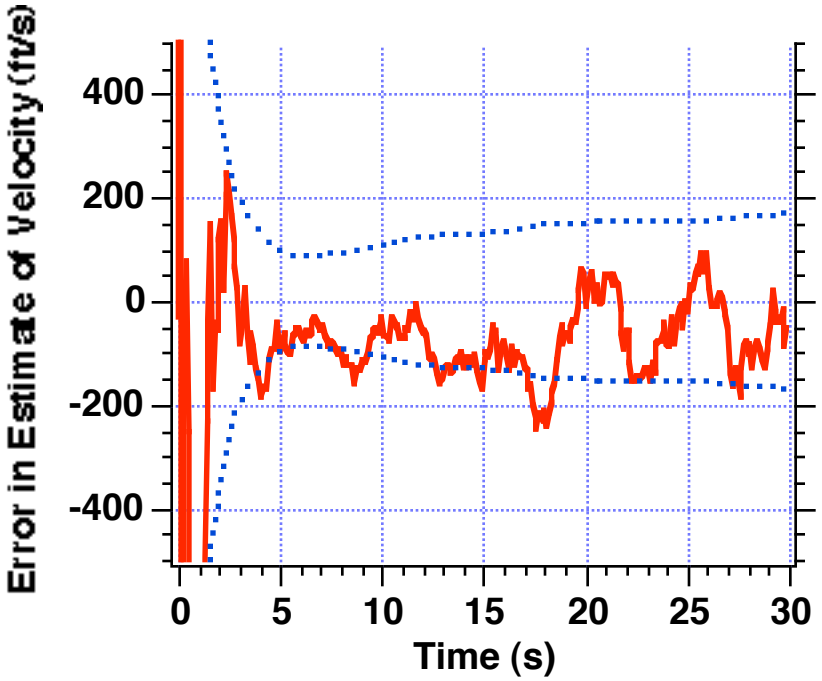
Adaptive Scheme Solves Divergence Problem

$$\Phi_{s_0} = 0$$

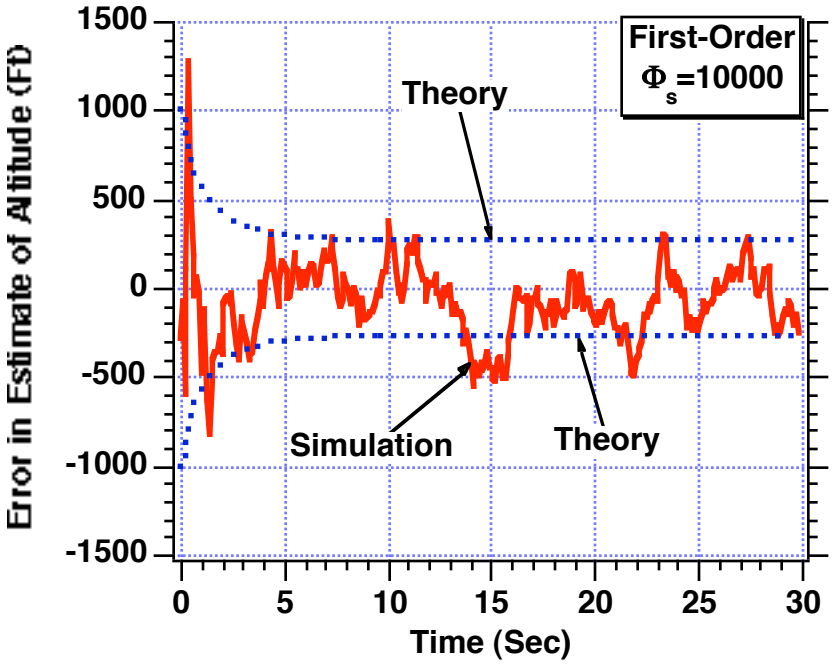
If (| Residual | > 2*Theory) Then

$$\Phi_{s_k} = \Phi_{s_{k-1}} + 1000$$





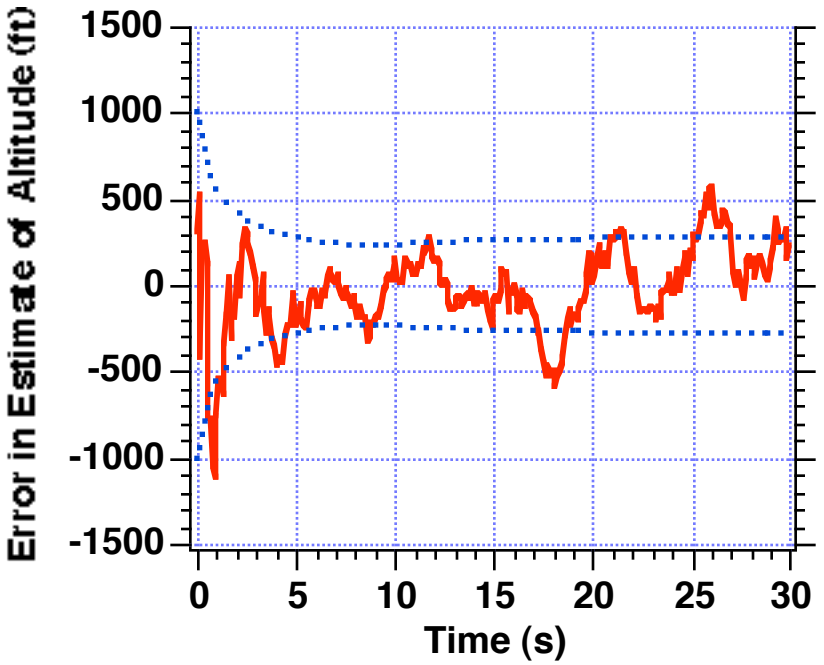
Adaptive Scheme Yields Comparable Results To Fixed Process Noise in Altitude



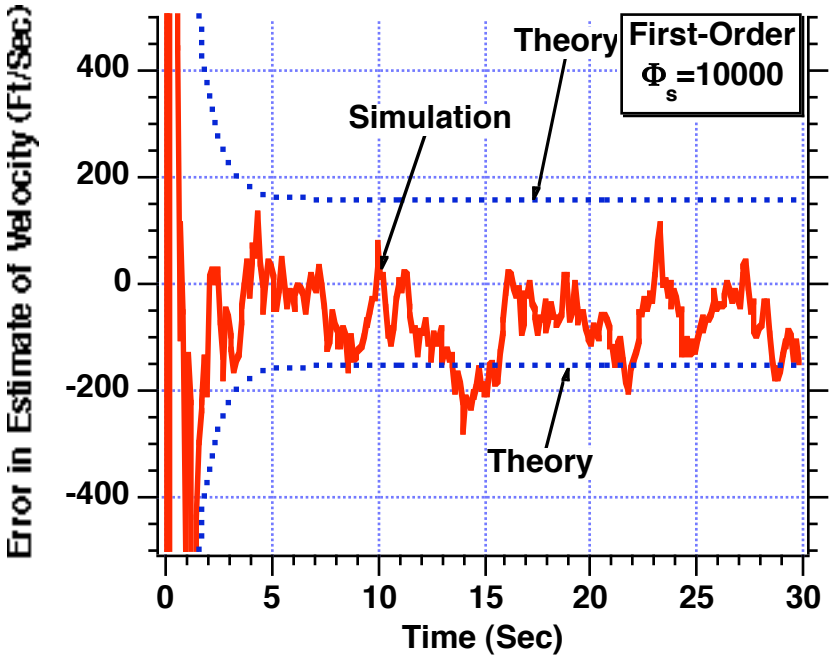
$\Phi_{s_0} = 0$

If (| Residual | > 2*Theory) Then

$\Phi_{s_k} = \Phi_{s_{k-1}} + 1000$



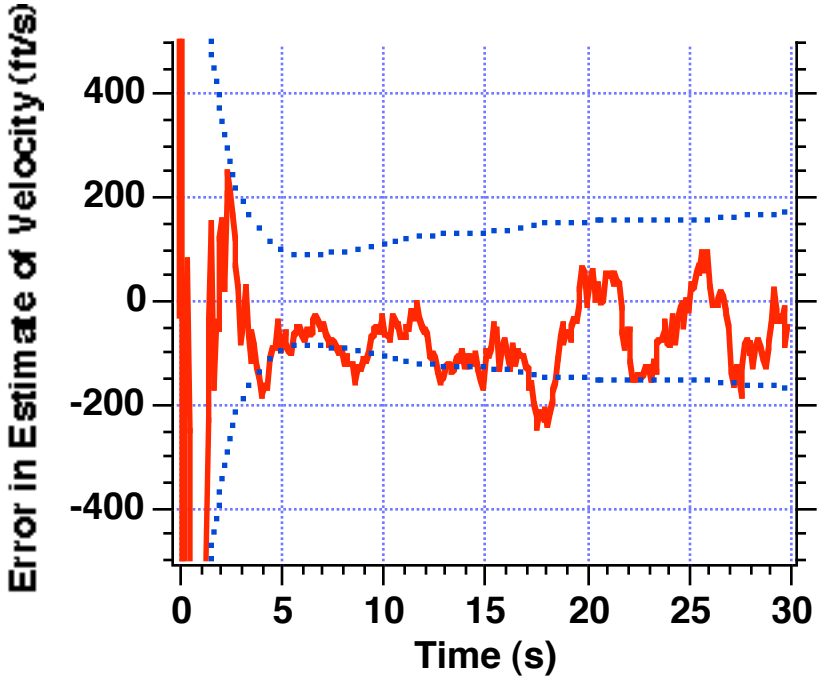
Adaptive Scheme Yields Comparable Results To Fixed Process Noise in Velocity



$$\Phi_{s_0} = 0$$

If (| Residual | > 2*Theory) Then

$$\Phi_{s_k} = \Phi_{s_{k-1}} + 1000$$

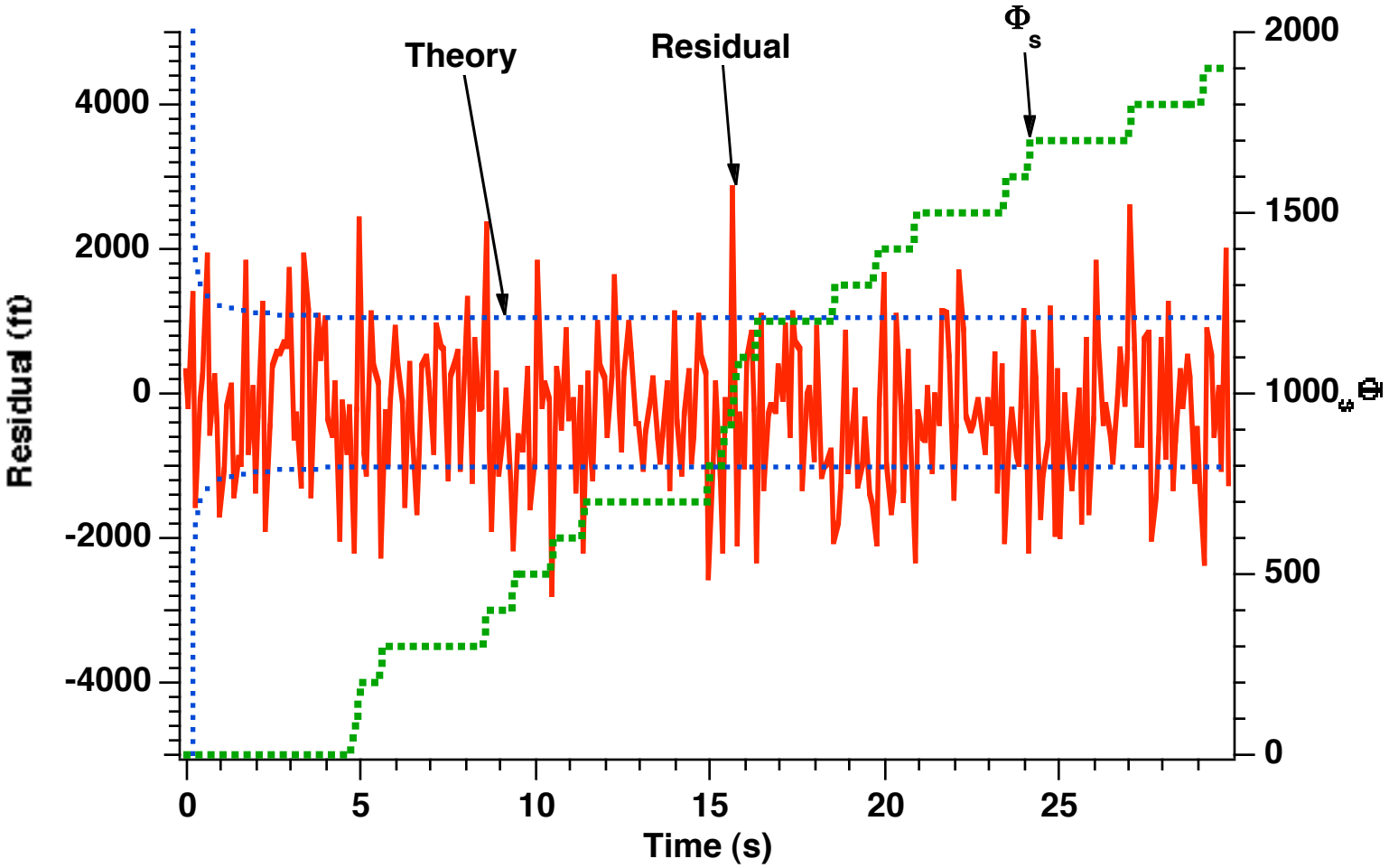


If Process Noise is Incremented by Smaller Amount it Changes More Often

$\Phi_{s_0} = 0$

If (| Residual | > 2*Theory) Then

$\Phi_{s_k} = \Phi_{s_{k-1}} + 100$

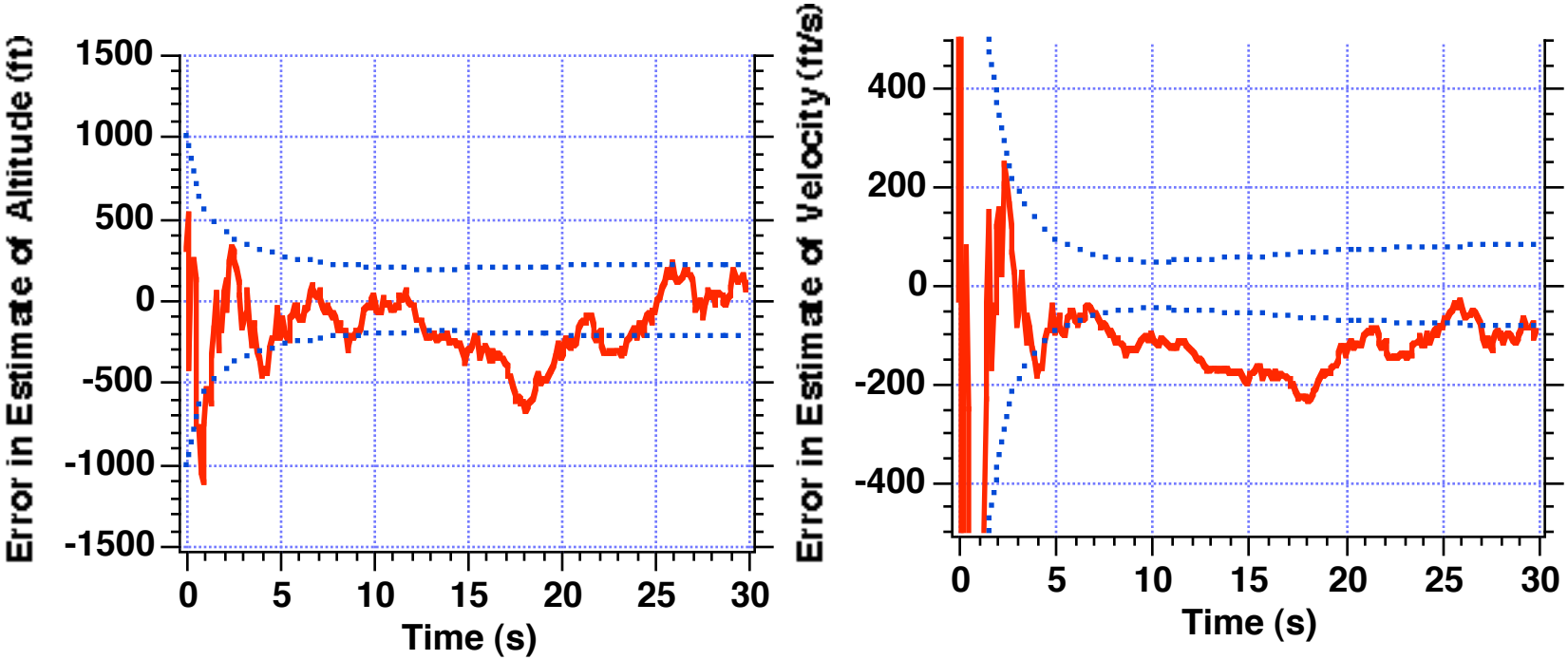


When Increment is Too Small There is Some Divergence in Velocity For a While

$$\Phi_{s_0} = 0$$

If (| Residual | > 2*Theory) Then

$$\Phi_{s_k} = \Phi_{s_{k-1}} + 100$$

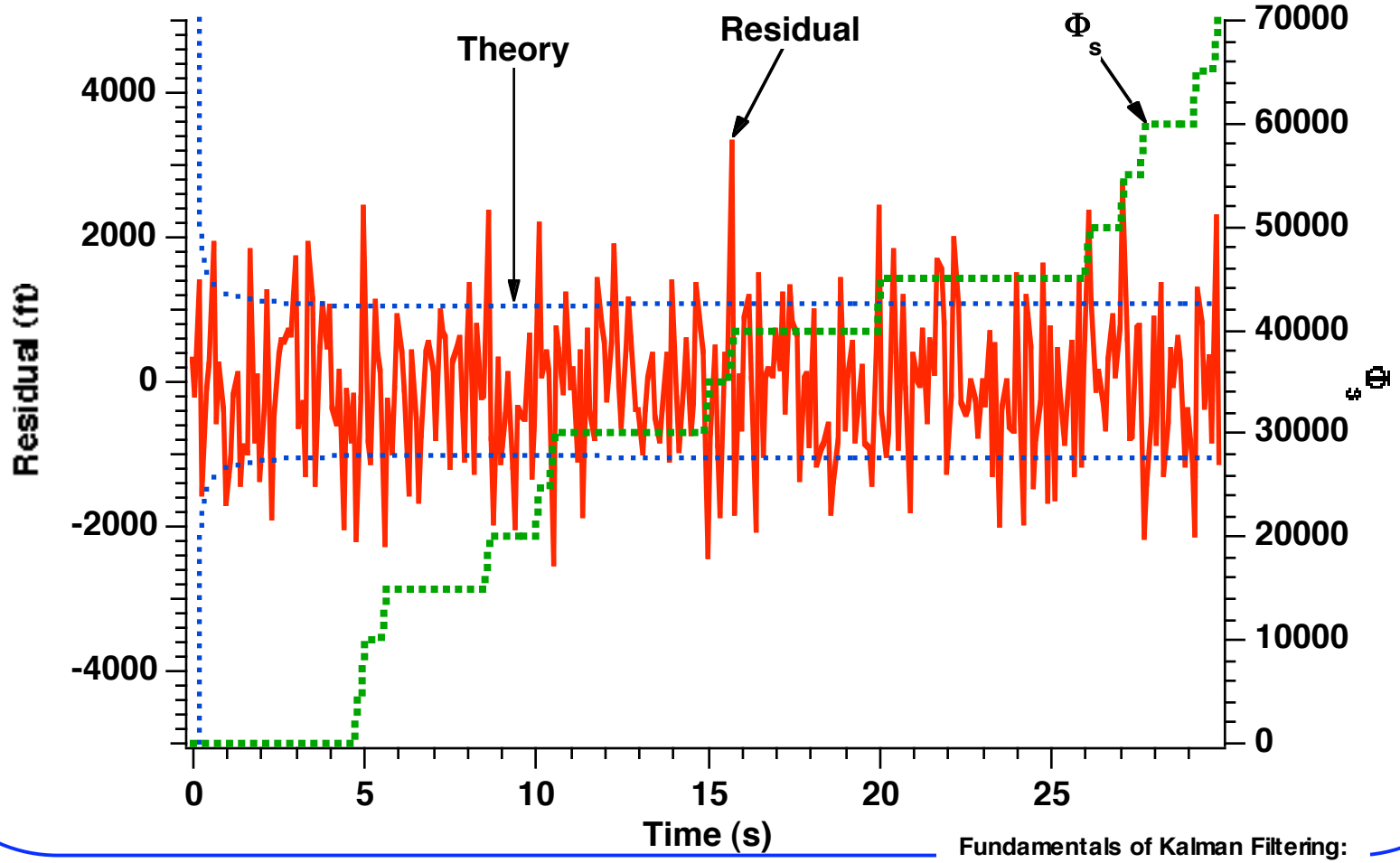


If Process Noise is Incremented by Too Large an Amount it Gets Very Large

$\Phi_{s_0} = 0$

If (| Residual | > 2*Theory) Then

$\Phi_{s_k} = \Phi_{s_{k-1}} + 5000$

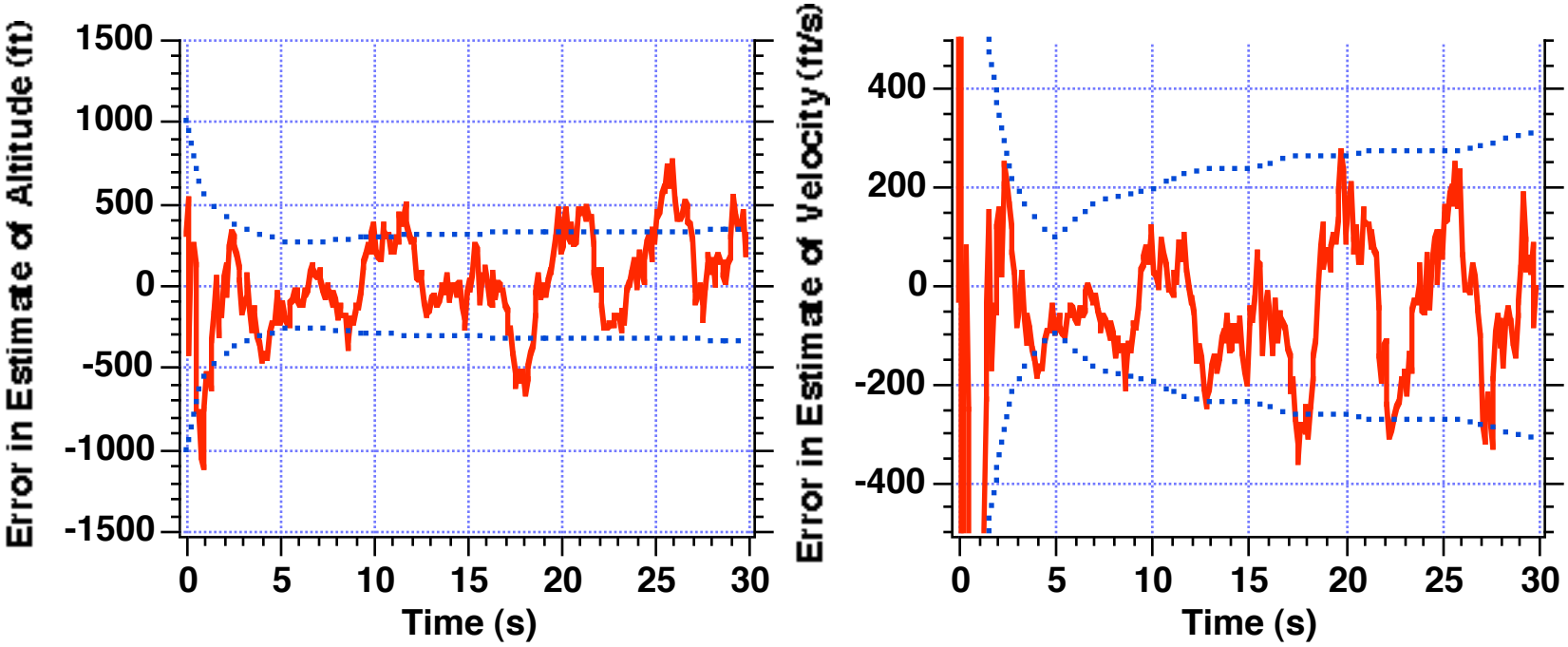


When Increment is Too Large The Errors in the Estimates Increase Because of More Process Noise

$$\Phi_{s_0} = 0$$

If (| Residual | > 2*Theory) Then

$$\Phi_{s_k} = \Phi_{s_{k-1}} + 5000$$

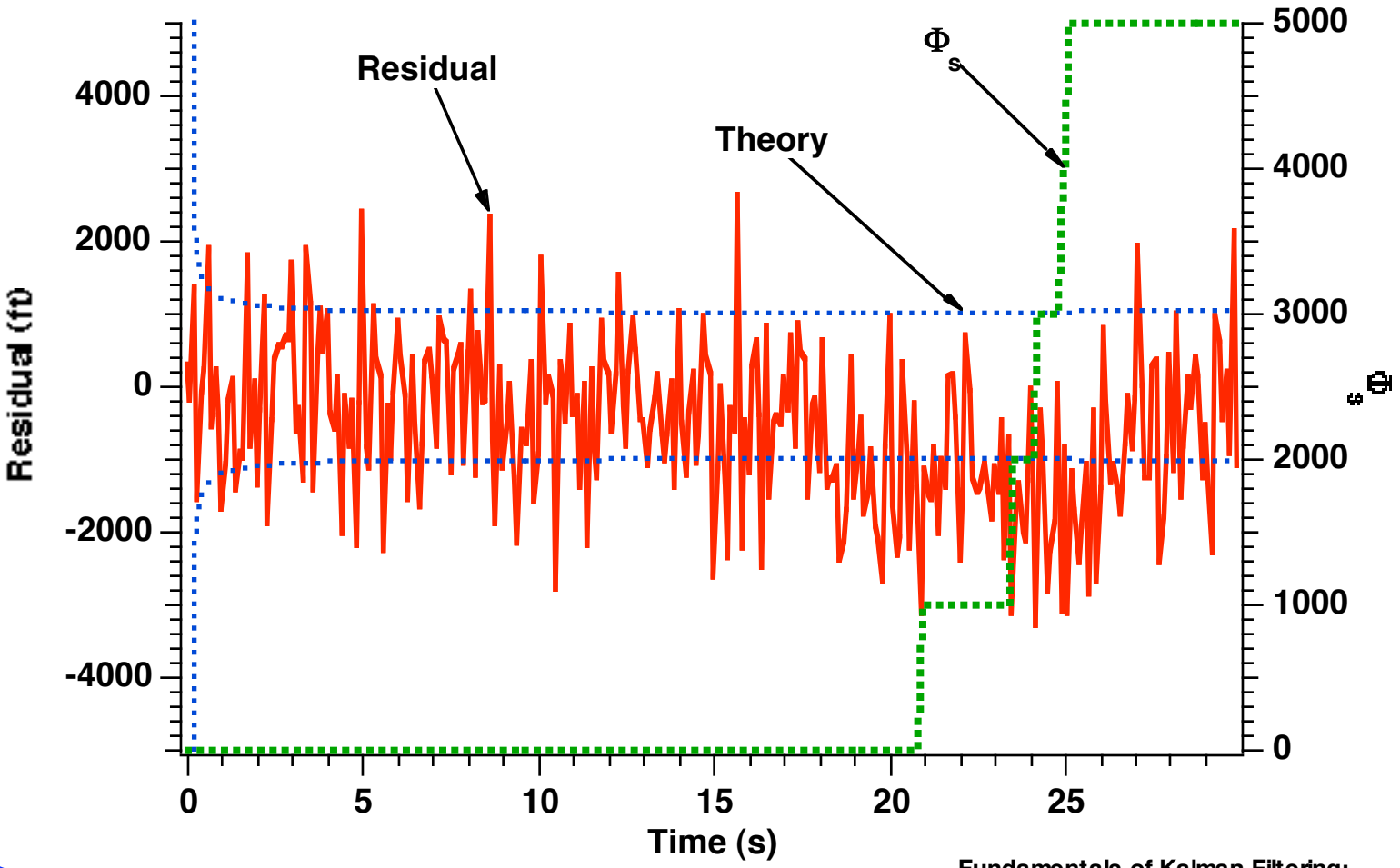


If Threshold is Too High The Process Noise Will Not Increase Enough

$\Phi_{s_0} = 0$

If (| Residual | > 3*Theory) Then

$\Phi_{s_k} = \Phi_{s_{k-1}} + 1000$



Insufficient Process Noise Leads To Divergence

$$\Phi_{s_0} = 0$$

If (| Residual | > 3*Theory) Then

$$\Phi_{s_k} = \Phi_{s_{k-1}} + 1000$$