Miscellaneous Topics on Measurements

Miscellaneous Topics on Measurements
Overview

- **Using additional measurements**
- **Asynchronous data rate**
- **Batch processing**

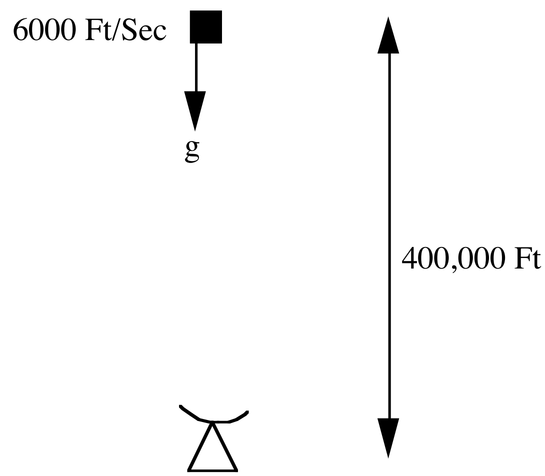
Using Additional Measurements

Using Additional Measurements
Overview

- Review of one measurement problem for tracing a falling object
    - Polynomial Kalman filter design with gravity compensation
- Incorporating a second measurement
    - New Kalman filter design
    - How to build filter when second measurement is at a different data rate

Review of One Measurement Problem For Tracking a Falling Object

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

Filter Design Making Use of A Priori Information - 1

Model of the real world

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \mathbf{w}$$

Kalman filter

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

Where

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

In our problem we know the model and measurement equation

$$\ddot{x} = -g \qquad x^* = x + v$$

Expressing gravitational information is state space form

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

By inspection

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \longrightarrow \boldsymbol{\Phi}_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Filter Design Making Use of A Priori Information - 2

Recall

$$\dot{\mathbf{x}} = \mathbf{Fx} + \mathbf{Gu} + \mathbf{w} \longrightarrow \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} g$$

We can also see that

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

Therefore

$$\mathbf{G_k} = \int_0^{T_s} \mathbf{\Phi}(\tau)\mathbf{G}\, \mathbf{d}\tau = \int_0^{T_s} \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} \mathbf{d}\tau = \begin{bmatrix} \dfrac{-T_s^2}{2} \\ -T_s \end{bmatrix}$$

Since formula of Kalman filter is given by

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

Substitution yields

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

Filter Design Making Use of A Priori Information - 3

Multiplying out the terms

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

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

If we define the residual as

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

The Kalman filter simplifies to

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

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

Riccati Equations

Riccati equations with process noise

$$\mathbf{M_k = \Phi_k P_{k-1} \Phi_k^T + Q_k} \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix} \qquad \mathbf{\Phi_k} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$$

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

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

Ramp Signal

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ u_s \end{bmatrix} \longrightarrow \mathbf{Q} = E\left[\begin{bmatrix} 0 \\ u_s \end{bmatrix} \begin{bmatrix} 0 & u_s \end{bmatrix}\right] = \begin{bmatrix} 0 & 0 \\ 0 & \Phi_s \end{bmatrix}$$

Measurement Noise Matrix is Scalar

$$\mathbf{R}_k = \sigma_p^2$$

Deriving discrete process noise matrix

$$\mathbf{Q_k} = \int_0^{T_s} \mathbf{\Phi(\tau) Q \Phi^T(\tau) dt}$$

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

MATLAB Version of First-Order Polynomial Kalman Filter with Gravity Compensation-1

```
TS=.1;
PHIS=0.;
A0=400000.;
A1=-6000.;
A2=-16.1;
XH=0.;
XDH=0.;
SIGNOISE=1000.;
ORDER=2;
T=0.;
S=0.;
H=.001;
PHI=[1 TS ;0 1];
P=[99999999 0;0 999999999];
IDNP=eye(ORDER);
Q=zeros(ORDER);
RMAT=SIGNOISE^2;
Q(1,1)=TS*TS*TS*PHIS/3.;
Q(1,2)=.5*TS*TS*PHIS;
Q(2,1)=Q(1,2);
Q(2,2)=PHIS*TS;
HMAT=[1 0];
HT=HMAT';
PHIT=PHI';
count=0;
for T=0:TS:30
        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 (PHI, P)

Process noise and measurement matrices (Q(1,1) through HMAT)

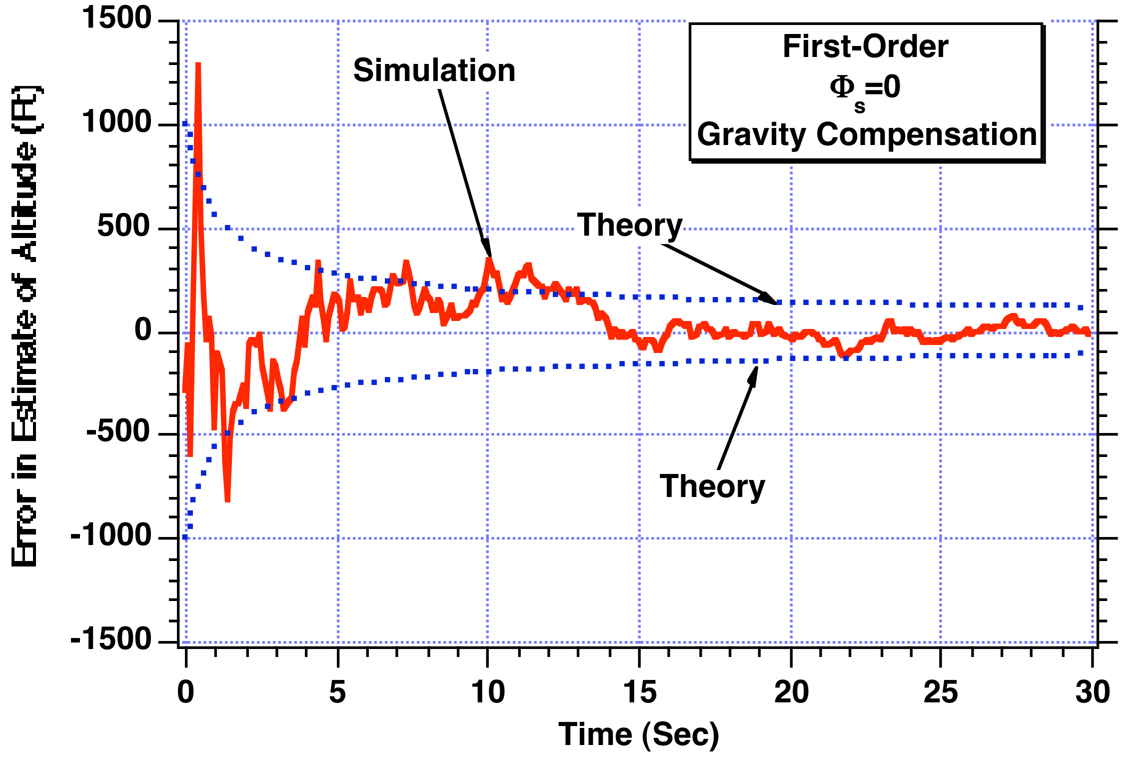
Riccati equations (PHIP through P=IKH*M)

MATLAB Version of First-Order Polynomial Kalman Filter with Gravity Compensation-2

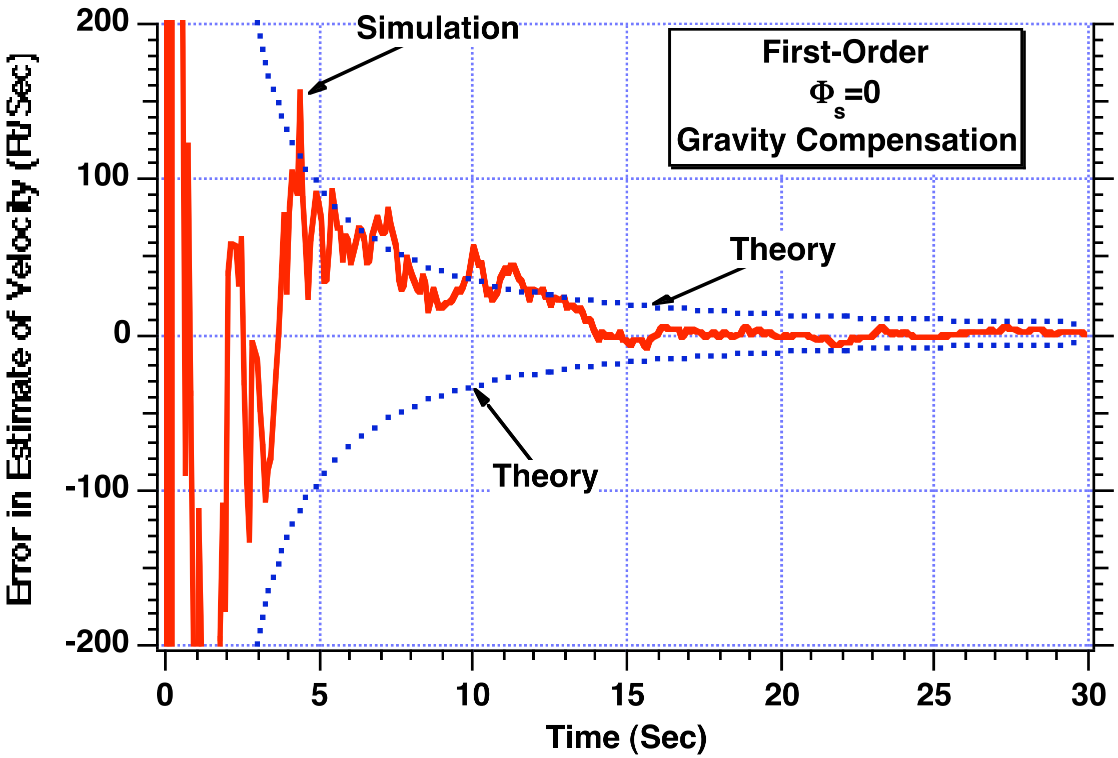
Filter has gravity compensation

```
        XNOISE=SIGNOISE*randn;
        X=A0+A1*T+A2*T*T;
        XD=A1+2*A2*T;
        XS=X+XNOISE;
        RES=XS-XH-TS*XDH+16.1*TS*TS;
        XH=XH+XDH*TS-16.1*TS*TS+K(1,1)*RES;
        XDH=XDH-32.2*TS+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
```

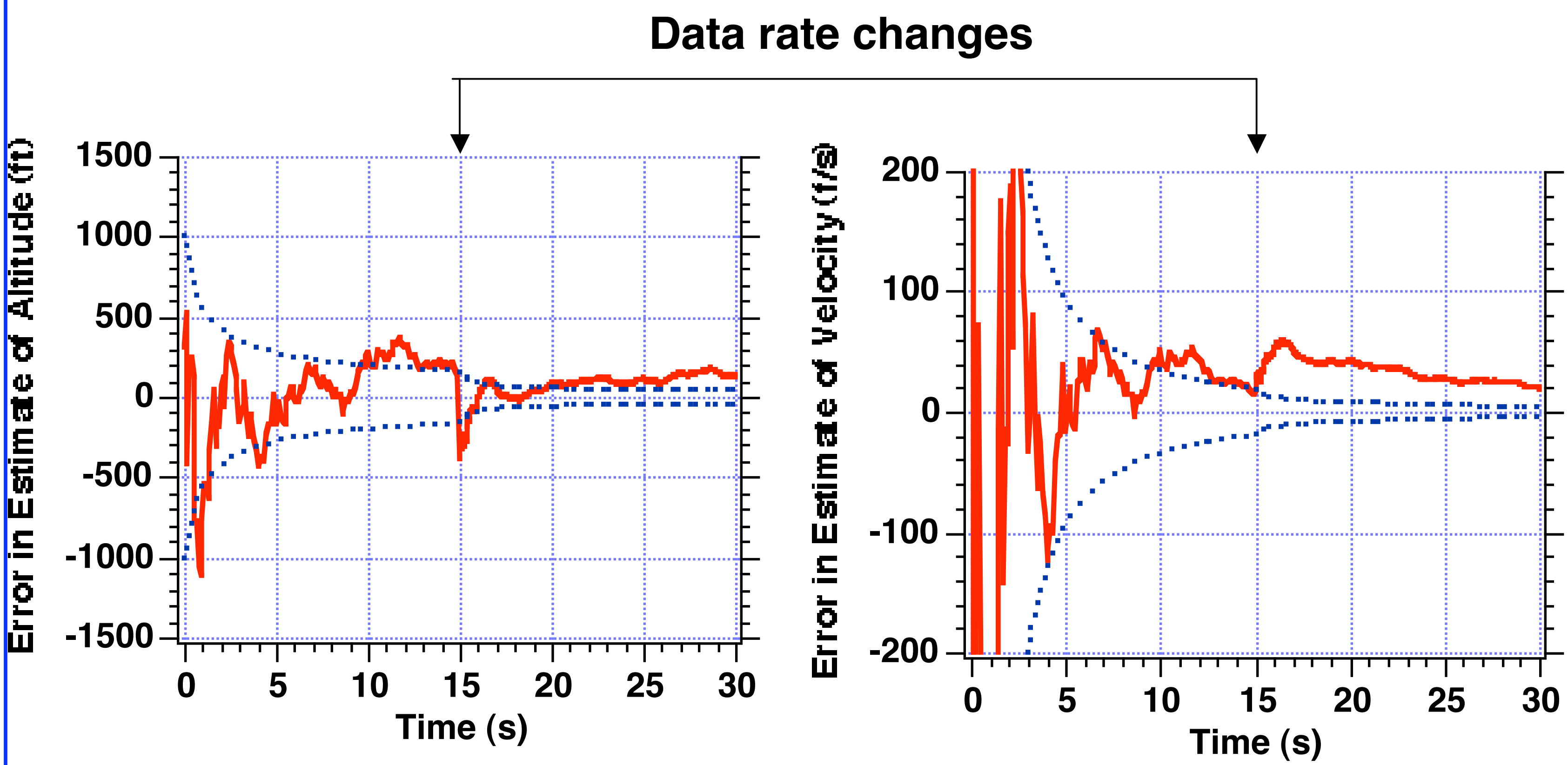
Typical Altitude Error Results For Case in Which There Are Only Position Measurements



Typical Velocity Error Results For Case in Which There Are Only Position Measurements



Going To Higher Data Rate During Flight Caused Transient Problem



T_s = 0.1 s for first 15 s and then T_s = 0.01 for last 15 s

Multiple Data Rate Simulation-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
      TS=.1
      PHIS=0.
      A0=400000.
      A1=-6000.
      A2=-16.1
      XH=0.
      XDH=0.
      SIGNOISE=1000.
      ORDER=2
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      T=0.
      S=0.
      H=.001
      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
```

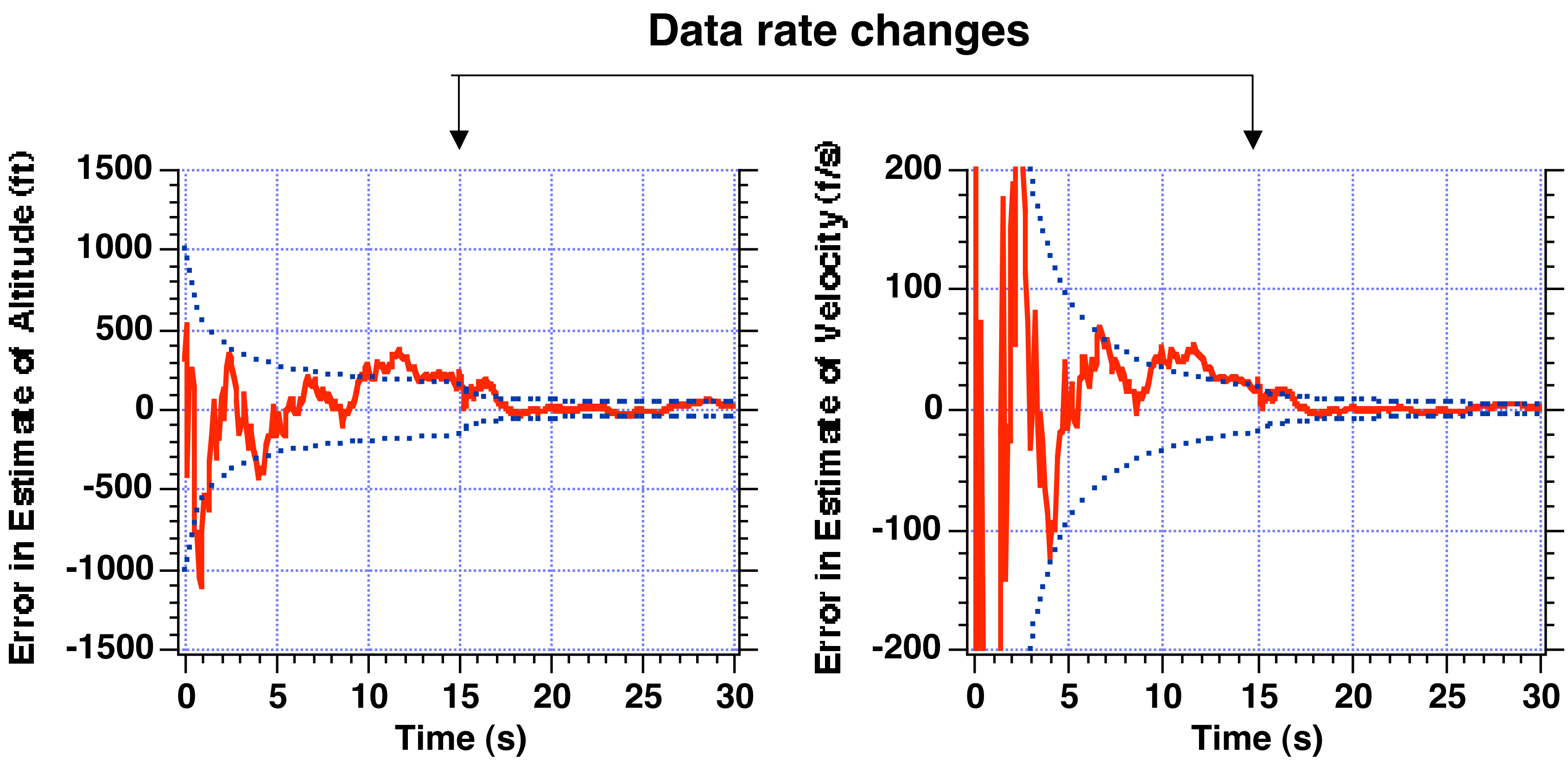
Multiple Data Rate Simulation-2

```
     RMAT(1,1)=SIGNOISE**2
     IDN(1,1)=1.
     IDN(2,2)=1.
     P(1,1)=99999999999.
     P(2,2)=99999999999.

     HMAT(1,1)=1.
     HMAT(1,2)=0.
     T=0.
10   IF(T>30.)GOTO 999

     IF(T<15.)THEN
          TS=.1
          TSOLD=TS
     ELSE
          TS=.01
     ENDIF
     PHI(1,1)=1.
     PHI(1,2)=TSOLD
     PHI(2,2)=1.
     Q(1,1)=TSOLD*TSOLD*TSOLD*PHIS/3.
     Q(1,2)=.5*TSOLD*TSOLD*PHIS
     Q(2,1)=Q(1,2)
     Q(2,2)=PHIS*TSOLD
     CALL MATTRN(PHI,ORDER,ORDER,PHIT)
     CALL MATTRN(HMAT,1,ORDER,HT)
     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)
```

Multiple Data Rate Simulation-3

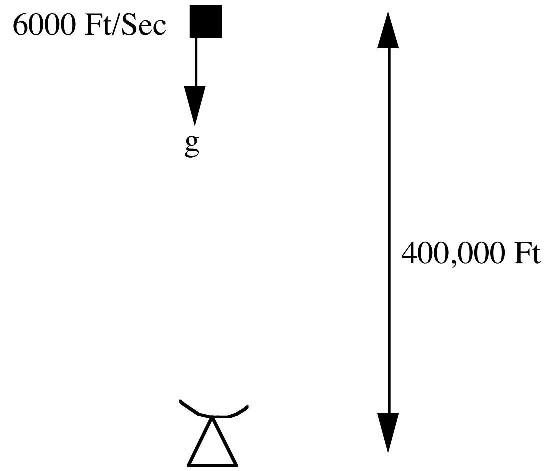
```
        HMHTRINV(1,1)=1./HMHTR(1,1)
        CALL MATMUL(M,ORDER,ORDER,HT,ORDER,1,MHT)
        CALL MATMUL(MHT,ORDER,1,HMHTRINV,1,1,K)
        CALL MATMUL(K,ORDER,1,HMAT,1,ORDER,KH)
        CALL MATSUB(IDN,ORDER,ORDER,KH,IKH)
        CALL MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
        CALL GAUSS(XNOISE,SIGNOISE)
        X=A0+A1*T+A2*T*T
        XD=A1+2*A2*T
        XS=X+XNOISE
        RES=XS-XH-TSOLD*XDH+16.1*TSOLD*TSOLD
        XH=XH+XDH*TSOLD-16.1*TSOLD*TSOLD+K(1,1)*RES
        XDH=XDH-32.2*TSOLD+K(2,1)*RES
        SP11=SQRT(P(1,1))
        SP22=SQRT(P(2,2))
        XHERR=X-XH
        XDHERR=XD-XDH
        WRITE(9,*)T,XD,XDH,K(1,1),K(2,1)
        WRITE(1,*)T,X,XH,XD,XDH
        WRITE(2,*)T,XHERR,SP11,-SP11,XDHERR,SP22,-SP22
        T=T+TS
        TSOLD=TS
        GOTO 10
999     CONTINUE
        PAUSE
        CLOSE(1)
        END
```

Adjusting Fundamental Matrix Allows Us To Change Data Rates Without Transient Effects



Incorporating a Second Measurement

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

Radar measures velocity with standard deviation of 10 ft/s

Desire to track object and estimate altitude and velocity

Filter Design With Two Measurements - 1

Model of the real world

$$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{u} + \mathbf{w}$$

Kalman filter

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

Where

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

In our problem we know the model and measurement equation

$$\ddot{x} = -g \qquad x^* = x + v_1$$

$$\dot{x}^* = \dot{x} + v_2$$

Expressing gravitational information is state space form

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} g \qquad \begin{bmatrix} x^* \\ \dot{x}^* \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

By inspection

$$\mathbf{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \longrightarrow \Phi_k = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Filter Design With Two Measurements - 2

Recall

$$\dot{\mathbf{x}} = \mathbf{Fx} + \mathbf{Gu} + \mathbf{w} \longrightarrow \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix} g$$

We can also see that

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

Therefore

$$\mathbf{G_k} = \int_0^{T_s} \Phi(\tau)\mathbf{G}\, d\tau = \int_0^{T_s} \begin{bmatrix} 1 & \tau \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix} d\tau = \begin{bmatrix} \frac{-T_s^2}{2} \\ -T_s \end{bmatrix}$$

Since formula of Kalman filter is given by

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

Substitution yields

$$\begin{bmatrix} \hat{x}_k \\ \hat{\dot{x}}_k \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \end{bmatrix} + \begin{bmatrix} -.5T_s^2 \\ -T_s \end{bmatrix} g + \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \left\{ \begin{bmatrix} x^* \\ \dot{x}^* \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{k-1} \\ \hat{\dot{x}}_{k-1} \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -.5T_s^2 \\ -T_s \end{bmatrix} g \right\}$$

Filter Design With Two Measurements - 3

Multiplying out the terms

$$\hat{x}_k = \hat{x}_{k-1} + T_s\hat{\dot{x}}_{k-1} - .5gT_s^2 + K_{11}\left(x_k^* - \hat{x}_{k-1} - T_s\hat{\dot{x}}_{k-1} + .5gT_s^2\right) + K_{12}\left(\dot{x}_k^* - \hat{\dot{x}}_{k-1} + gT_s\right)$$

$$\hat{\dot{x}}_k = \hat{\dot{x}}_{k-1} - gT_s + K_{21}\left(x_k^* - \hat{x}_{k-1} - T_s\hat{\dot{x}}_{k-1} + .5gT_s^2\right) + K_{22}\left(\dot{x}_k^* - \hat{\dot{x}}_{k-1} + gT_s\right)$$

If we define the residual as

$$RES_{1_k} = x_k^* - \hat{x}_{k-1} - T_s\hat{\dot{x}}_{k-1} + .5gT_s^2$$

$$RES_{2_k} = \dot{x}_k^* - \hat{\dot{x}}_{k-1} + gT_s$$

The Kalman filter simplifies to

$$\hat{x}_k = \hat{x}_{k-1} + T_s\hat{\dot{x}}_{k-1} - .5gT_s^2 + K_{11}RES_{1_k} + K_{12}RES_{2_k}$$

$$\hat{\dot{x}}_k = \hat{\dot{x}}_{k-1} - gT_s + K_{21}RES_{1_k} + K_{22}RES_{2_k}$$

Riccati Equations For Two Measurements

Riccati equations with process noise

$$\mathbf{M_k} = \mathbf{\Phi_k P_{k-1} \Phi_k^T} + \mathbf{Q_k} \qquad \mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \mathbf{\Phi_k} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}$$

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

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

Ramp Signal

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ u_s \end{bmatrix} \longrightarrow \mathbf{Q} = E\left[\begin{bmatrix} 0 \\ u_s \end{bmatrix} \begin{bmatrix} 0 & u_s \end{bmatrix}\right] = \begin{bmatrix} 0 & 0 \\ 0 & \Phi_s \end{bmatrix}$$

Measurement Noise Matrix Has Changed

$$\mathbf{R}_k = \begin{bmatrix} \sigma_p^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

Deriving discrete process noise matrix

$$\mathbf{Q_k} = \int_0^{T_s} \mathbf{\Phi(\tau) Q \Phi^T(\tau) dt}$$

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

FORTRAN Version of Two Measurement Filter-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(2,2),HT(2,2),PHIT(2,2)
      REAL*8 RMAT(2,2),IDN(2,2),PHIP(2,2),PHIPPHIT(2,2),HM(2,2)
      REAL*8 HMHT(2,2),HMHTR(2,2),HMHTRINV(2,2),MHT(2,2),K(2,2)
      REAL*8 KH(2,2),IKH(2,2)
      INTEGER ORDER
C     IF NSAMP=0 WE HAVE BOTH MEASUREMENTS AT THE SAME RATE
C     IF NSAMP=INFINITY WE ONLY HAVE POSITION MEASUREMENT
      NSAMP=5
      TS=.1
      PHIS=0.
      A0=400000.
      A1=-6000.
      A2=-16.1
      XH=0.
      XDH=0.
      SIGNOISE=1000.
      SIGVEL=10.
      ORDER=2
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      T=0.
      S=0.
      H=.001
      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.
      RMAT(I,J)=0.
      HMAT(I,J)=0.
14    CONTINUE
      RMAT(1,1)=SIGNOISE**2
      RMAT(2,2)=SIGVEL**2
```

Position and velocity measurements at different times

Measurement noise

Measurement noise matrix

FORTRAN Version of Two Measurement Filter-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.
Q(1,1)=TS*TS*TS*PHIS/3.
Q(1,2)=.5*TS*TS*PHIS
Q(2,1)=Q(1,2)
Q(2,2)=PHIS*TS
HMAT(1,1)=1.
N=0
DO 10 T=0.,30.,TS
IF(N>=NSAMP)THEN
            HMAT(2,2)=1.
ENDIF
CALL MATTRN(PHI,ORDER,ORDER,PHIT)
CALL MATTRN(HMAT,ORDER,ORDER,HT)
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,ORDER,ORDER,M,ORDER,ORDER,HM)
CALL MATMUL(HM,ORDER,ORDER,HT,ORDER,ORDER,HMHT)
CALL MATADD(HMHT,ORDER,ORDER,RMAT,HMHTR)
DET=HMHTR(1,1)*HMHTR(2,2)-HMHTR(1,2)*HMHTR(2,1)
HMHTRINV(1,1)=HMHTR(2,2)/DET
HMHTRINV(1,2)=-HMHTR(1,2)/DET
HMHTRINV(2,1)=-HMHTR(2,1)/DET
HMHTRINV(2,2)=HMHTR(1,1)/DET
CALL MATMUL(M,ORDER,ORDER,HT,ORDER,ORDER,MHT)
CALL MATMUL(MHT,ORDER,ORDER,HMHTRINV,ORDER,ORDER,K)
CALL MATMUL(K,ORDER,ORDER,HMAT,ORDER,ORDER,KH)
CALL MATSUB(IDN,ORDER,ORDER,KH,IKH)
CALL MATMUL(IKH,ORDER,ORDER,M,ORDER,ORDER,P)
```

Change measurement matrix when velocity Measurement is available

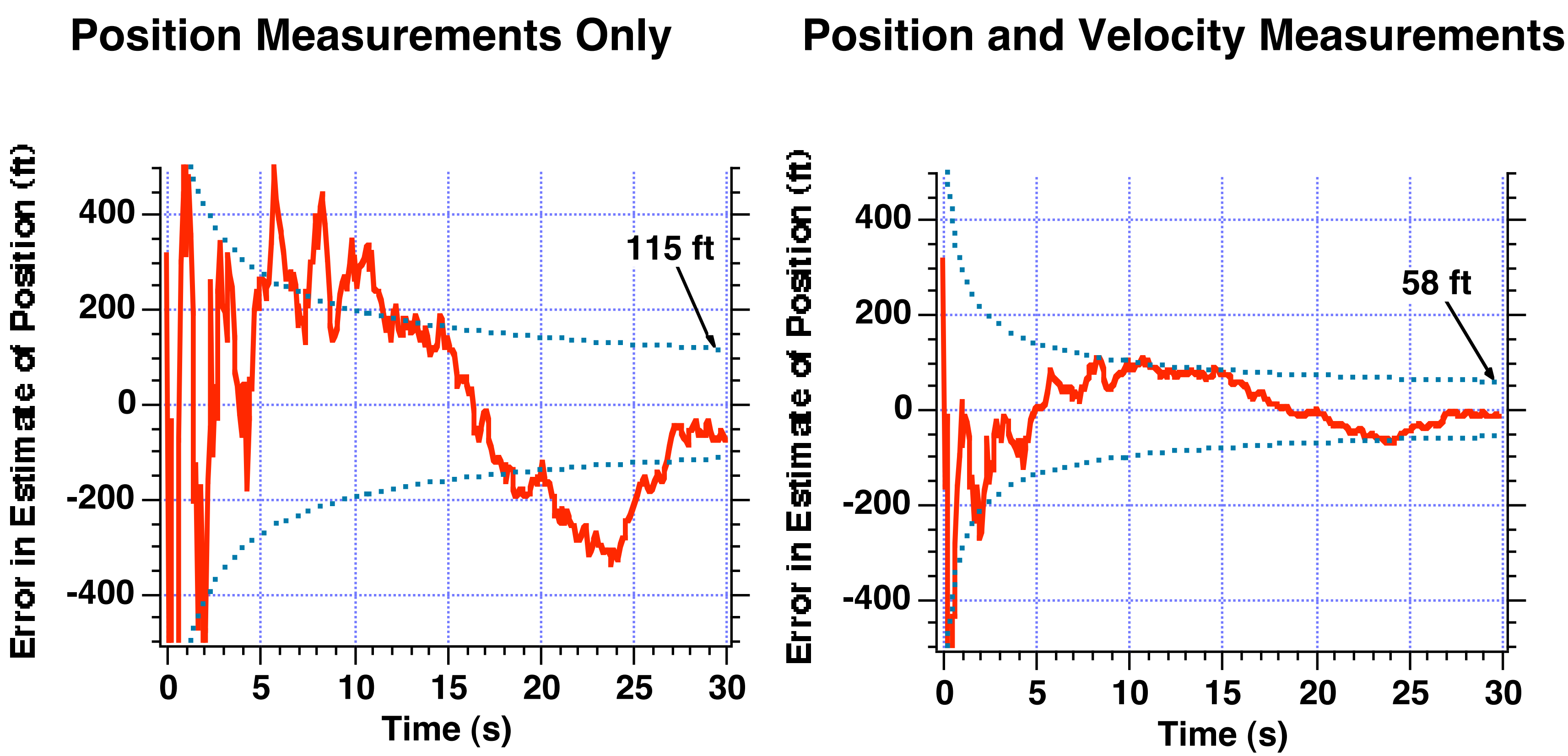
FORTRAN Version of Two Measurement Filter-3

```
      CALL GAUSS(XNOISE,SIGNOISE)
      CALL GAUSS(XDNOISE,SIGVEL)
      X=A0+A1*T+A2*T*T
      XD=A1+2*A2*T
      XS=X+XNOISE
      XDS=XD+XDNOISE
      RES1=XS-XH-TS*XDH+16.1*TS*TS
      RES2=0.
      IF(N>=NSAMP)THEN
            N=0
            RES2=XDS-XDH+32.2*TS
      ENDIF
      XH=XH+XDH*TS-16.1*TS*TS+K(1,1)*RES1+K(1,2)*RES2
      XDH=XDH-32.2*TS+K(2,1)*RES1+K(2,2)*RES2
      SP11=SQRT(P(1,1))
      SP22=SQRT(P(2,2))
      XHERR=X-XH
      XDHERR=XD-XDH

      WRITE(9,*)T,XD,XDH,K(1,1),K(2,1)
      WRITE(1,*)T,X,XH,XD,XDH
      WRITE(2,*)T,XHERR,SP11,-SP11,XDHERR,SP22,-SP22,HMAT(2,2),RES2
      N=N+1
      HMAT(2,2)=0.
10    CONTINUE
      PAUSE
      CLOSE(1)
      END
```

New residual when velocity measurement is available

Reset measurement matrix

Addition of Velocity Measurement Reduces Position Estimation Error By Factor of Two

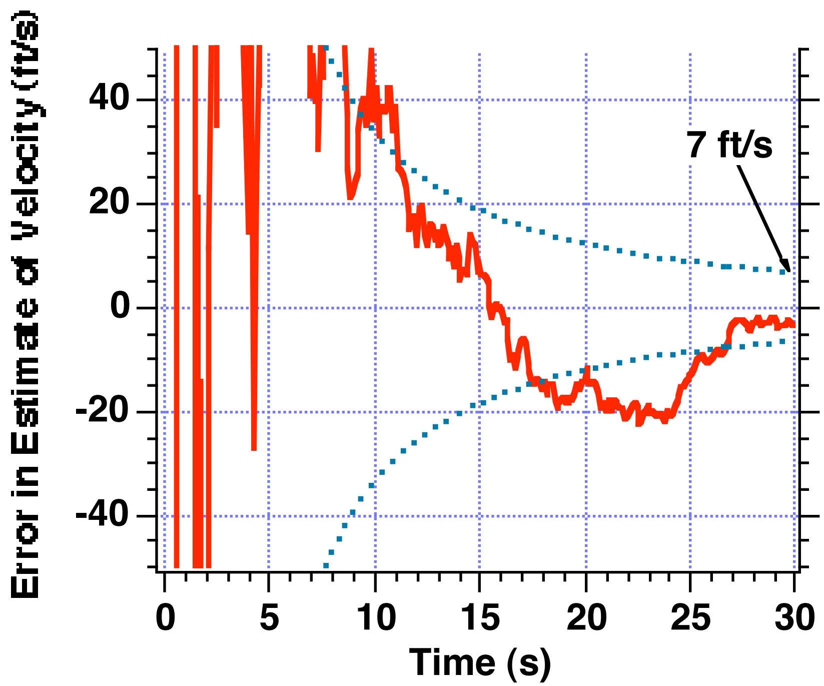


There are 10 position measurements per second

There are 10 position and velocity measurements per second

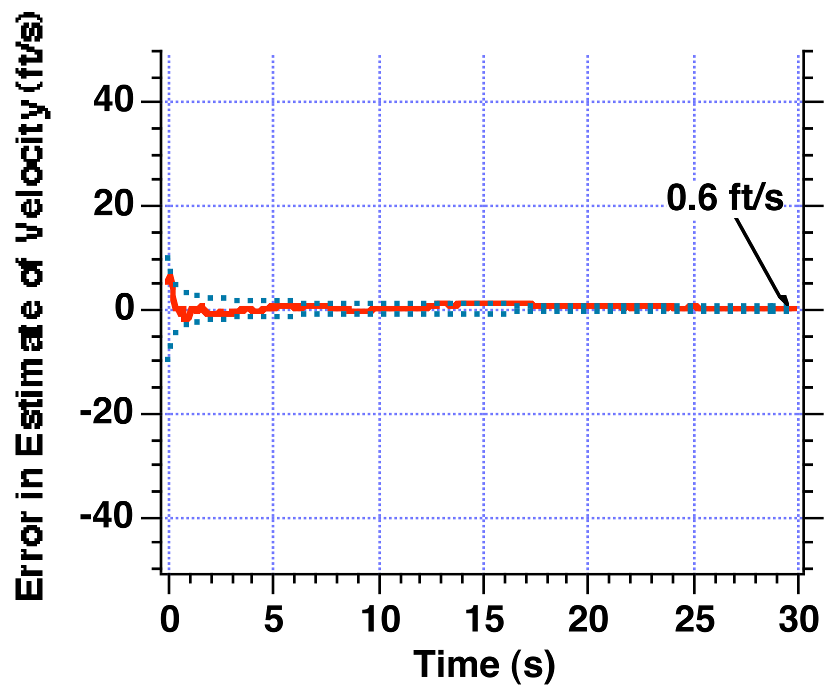
Addition of Velocity Measurement Reduces Velocity Estimation Error By Order of Magnitude

Position Measurements Only



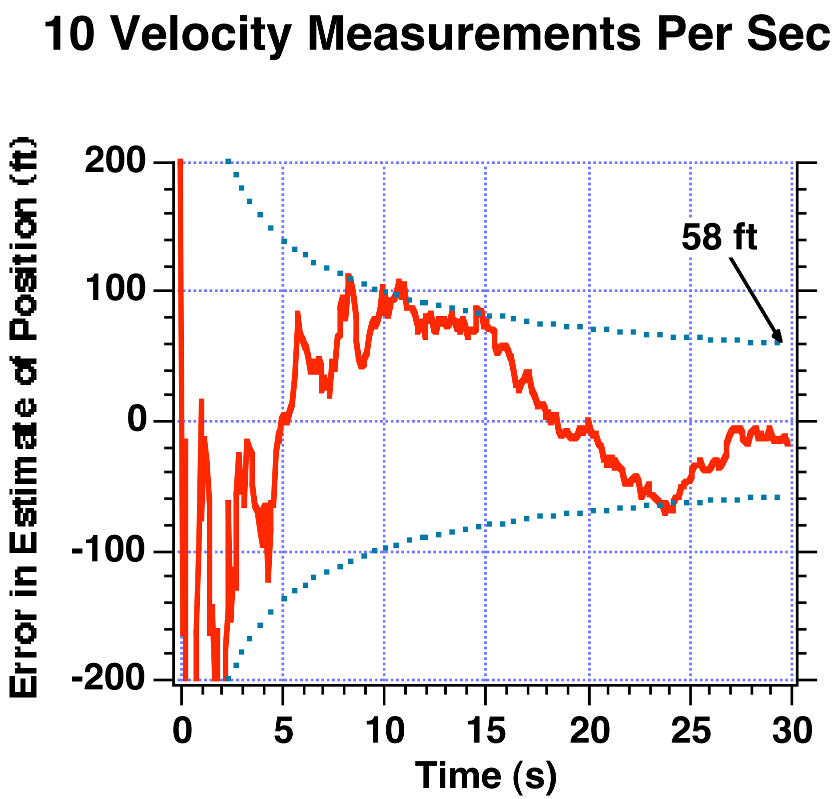
There are 10 position measurements per second

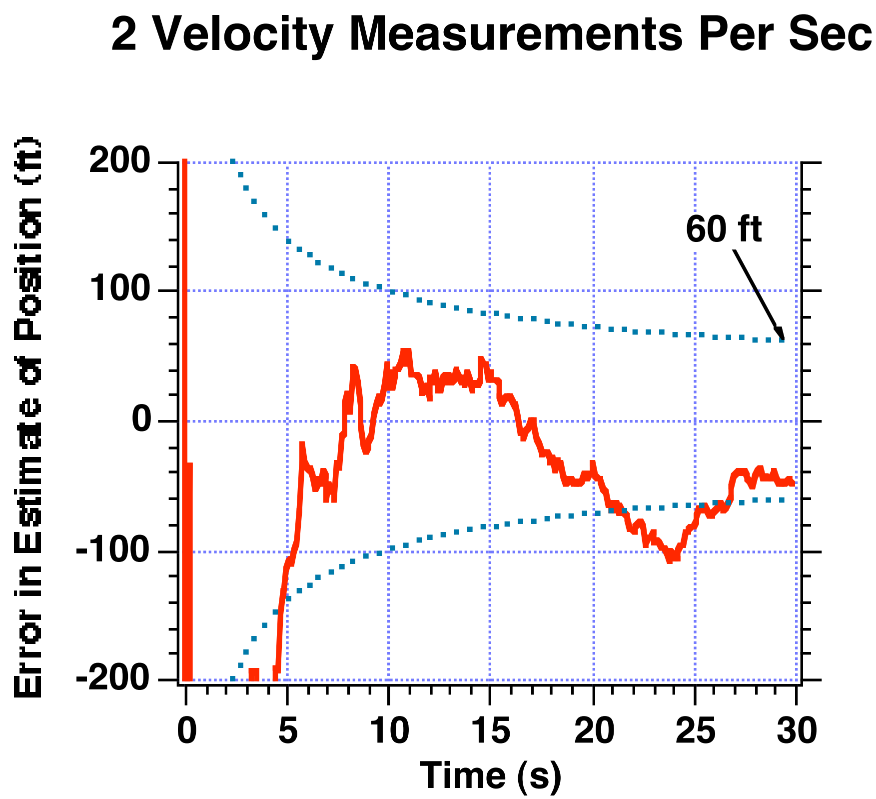
Position and Velocity Measurements



There are 10 position and velocity measurements per second

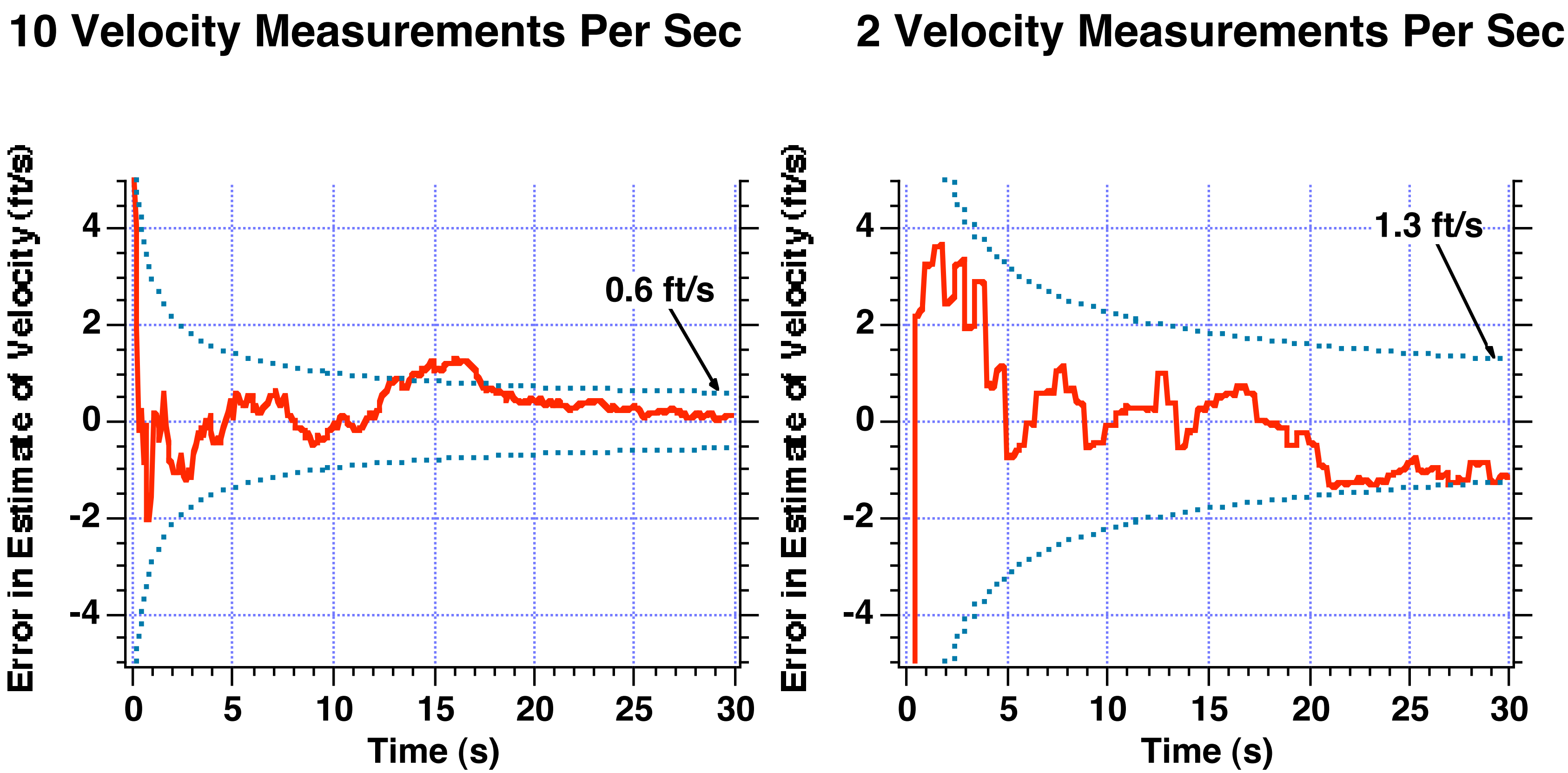
Having Fewer Velocity Measurements Slightly Increases Position Estimation Error





There are 10 position measurements per second

Having Fewer Velocity Measurements Increases Velocity Estimation Error



There are 10 position measurements per second

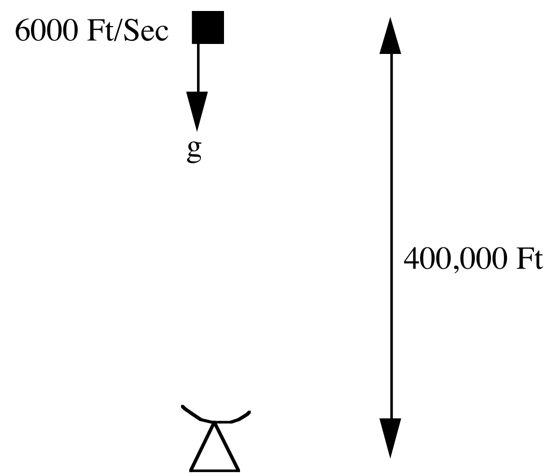
Using Additional Measurements
Summary

- Addition of second measurement significantly reduces estimation errors
- Modeling of second measurement at different data rate is easy

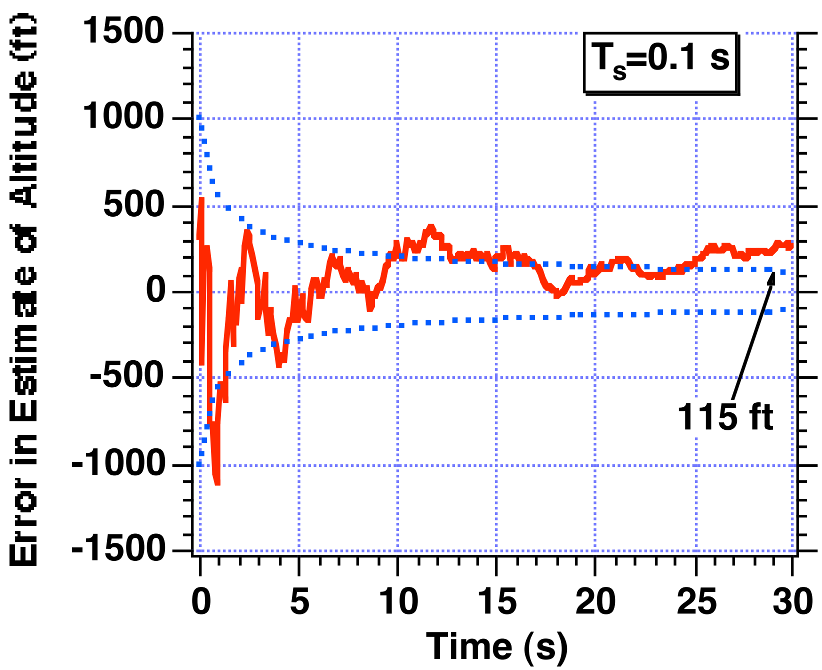
Asynchronous Data Rate

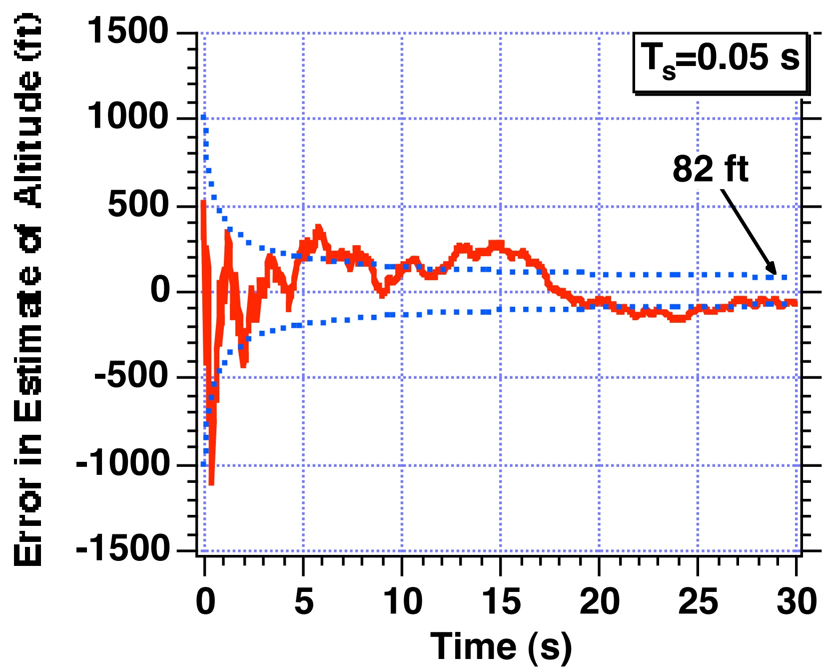
Problem

In some applications data rate varies and is not known in advance
We will revisit falling object problem of previous sections

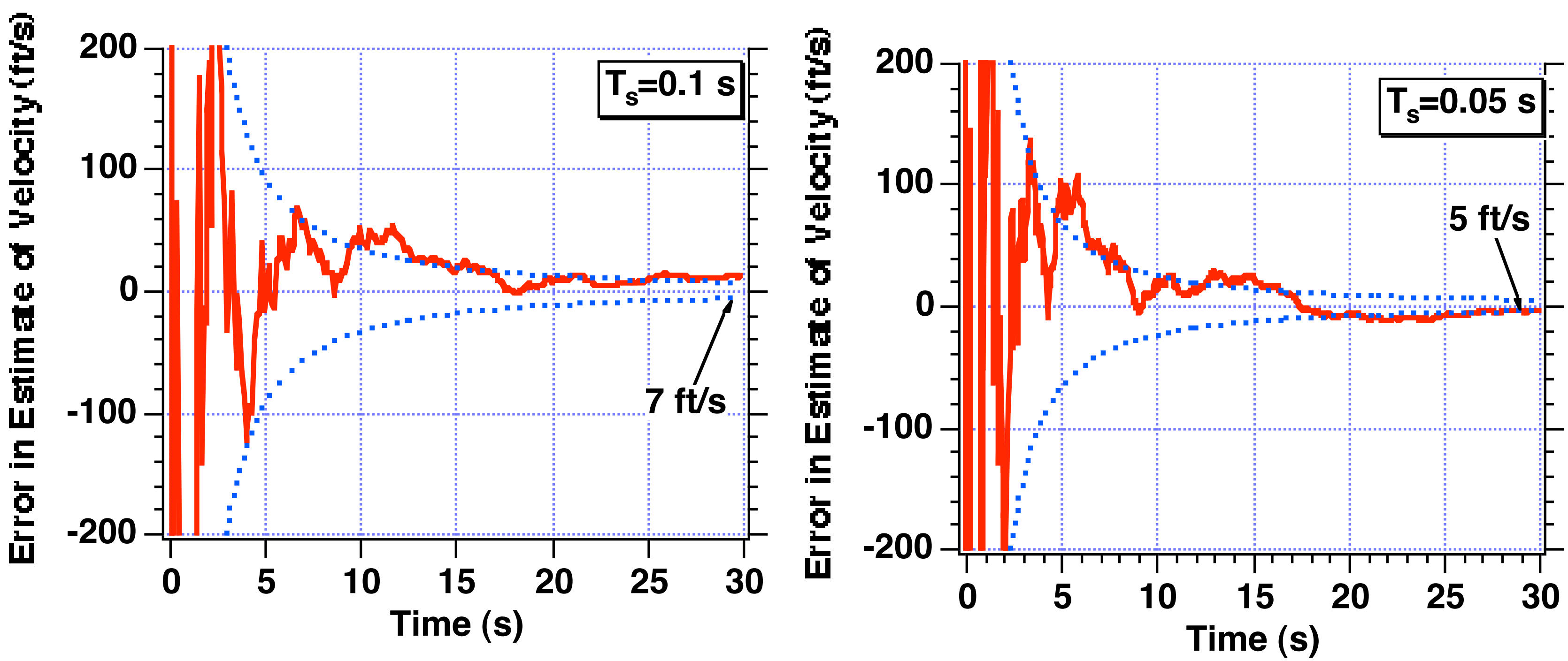


Error in Estimate of Altitude Improves If Fixed Sampling Time is Smaller





Error in Estimate of Velocity Improves If Fixed Sampling Time is Smaller



Asynchronous Kalman Filter-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
      LOGICAL QFIRST
      QFIRST=.TRUE.
      TS=.1
      PHIS=0.
      A0=400000.
      A1=-6000.
      A2=-16.1
      XH=0.
      XDH=0.
      SIGNOISE=1000.
      ORDER=2
      OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
      OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
      T=0.
      S=0.
      H=.001
      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
```

Needed for logic (LOGICAL QFIRST / QFIRST=.TRUE.)

Asynchronous Kalman Filter-2

```
        RMAT(1,1)=SIGNOISE**2
        IDN(1,1)=1.
        IDN(2,2)=1.
        P(1,1)=99999999999.
        P(2,2)=99999999999.
        HMAT(1,1)=1.
        HMAT(1,2)=0.
        T=0.
10      IF(T>30.)GOTO 999
        CALL GAUSS(XNOISE,SIGNOISE)
        X=A0+A1*T+A2*T*T
        XD=A1+2*A2*T
        XS=X+XNOISE
        CALL UNIF(XNOISE)
        TSP=XNOISE*.1
        IF(QFIRST)THEN
                XH=XS
                XDH=0.
                TOLD=T
                QFIRST=.FALSE.
        ELSE
                TS=T-TOLD
                PHI(1,1)=1.
                PHI(1,2)=TS
                PHI(2,2)=1.
                Q(1,1)=TS*TS*TS*PHIS/3.
                Q(1,2)=.5*TS*TS*PHIS
                Q(2,1)=Q(1,2)
                Q(2,2)=PHIS*TS
```

Model of real world comes first (X, XD)

Sampling time uniformly distributed Between 0 and .1 s (CALL UNIF, TSP)

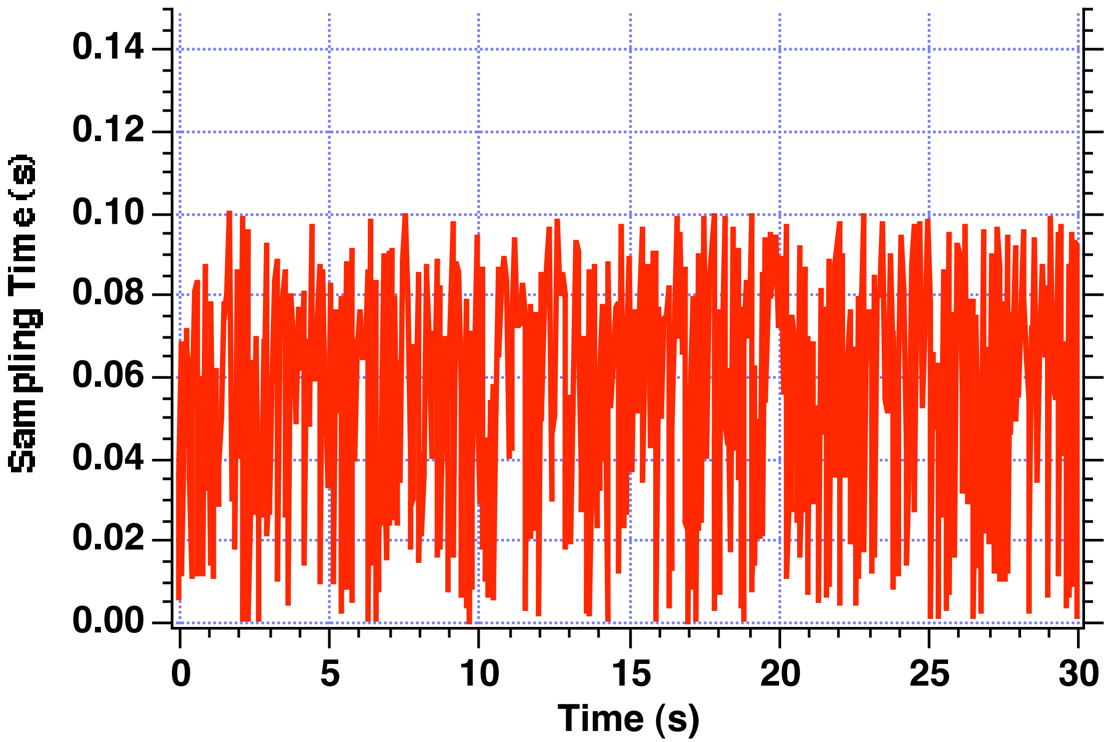
Initialize Kalman filter (XH through QFIRST)

Calculate new sampling time (TS=T-TOLD)

Asynchronous Kalman Filter-3

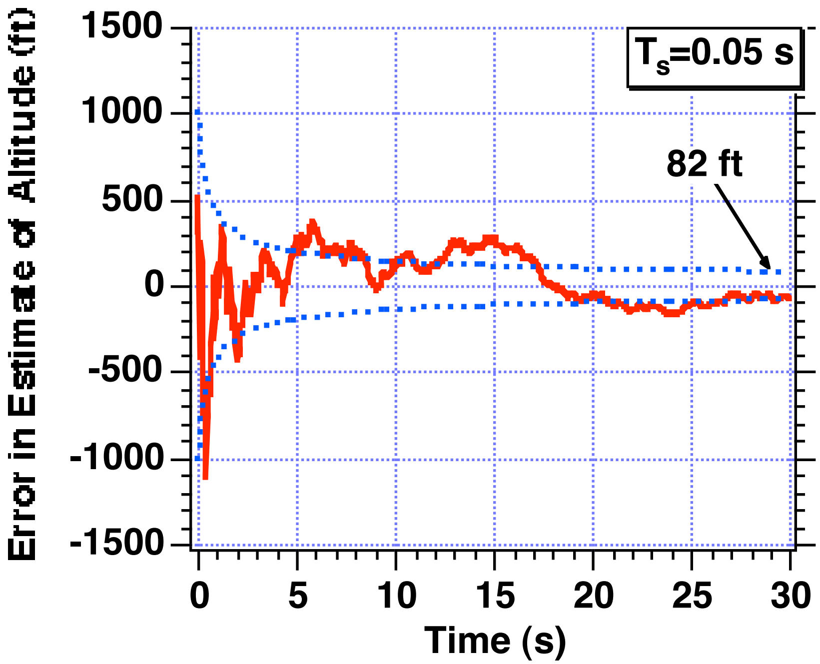
```
                    CALL MATTRN(PHI,ORDER,ORDER,PHIT)
                    CALL MATTRN(HMAT,1,ORDER,HT)
                    CALL MATMUL(PHI,ORDER,ORDER,P,ORDER,ORDER,PHIP)
                    CALL MATMUL(PHIP,ORDER,ORDER,PHIT,ORDER,ORDER,PHIPPHIT)
                    CALL MATADD(PHIPPHIT,ORDER,ORDER,Q,M)
                    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)
                    RES=XS-XH-TS*XDH+16.1*TS*TS
                    XH=XH+XDH*TS-16.1*TS*TS+K(1,1)*RES
                    XDH=XDH-32.2*TS+K(2,1)*RES
                    TOLD=T                  Reset
                    SP11=SQRT(P(1,1))
                    SP22=SQRT(P(2,2))
                    XHERR=X-XH
                    XDHERR=XD-XDH
                    WRITE(9,*)T,XD,XDH,TS,TSP
                    WRITE(1,*)T,X,XH,XD,XDH,TS,TSP
                    WRITE(2,*)T,XHERR,SP11,-SP11,XDHERR,SP22,-SP22
               ENDIF
               T=T+TSP           Update time
               GOTO 10
999            CONTINUE
               PAUSE
               CLOSE(1)
               END
```

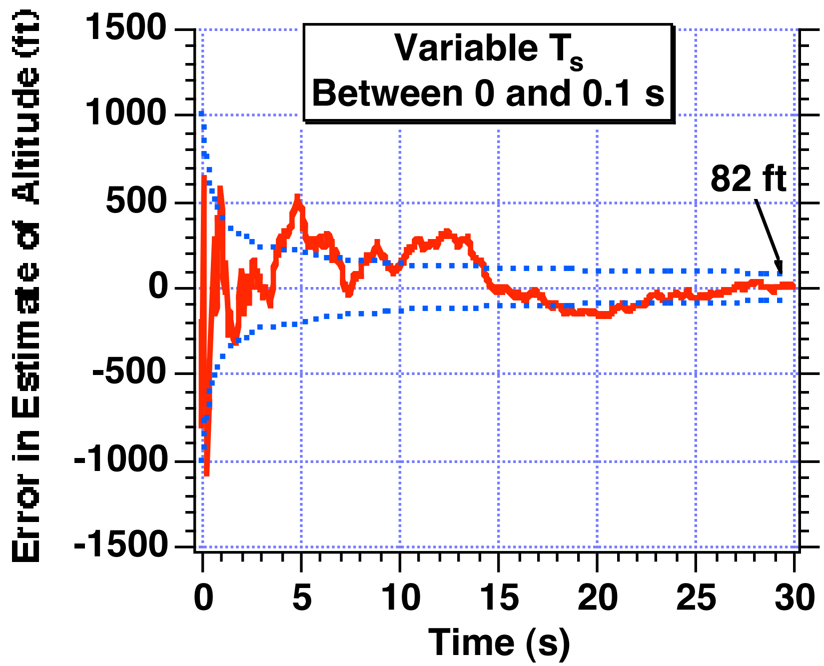
Sampling Time is Uniformly Distributed Between 0 and 0.1 s



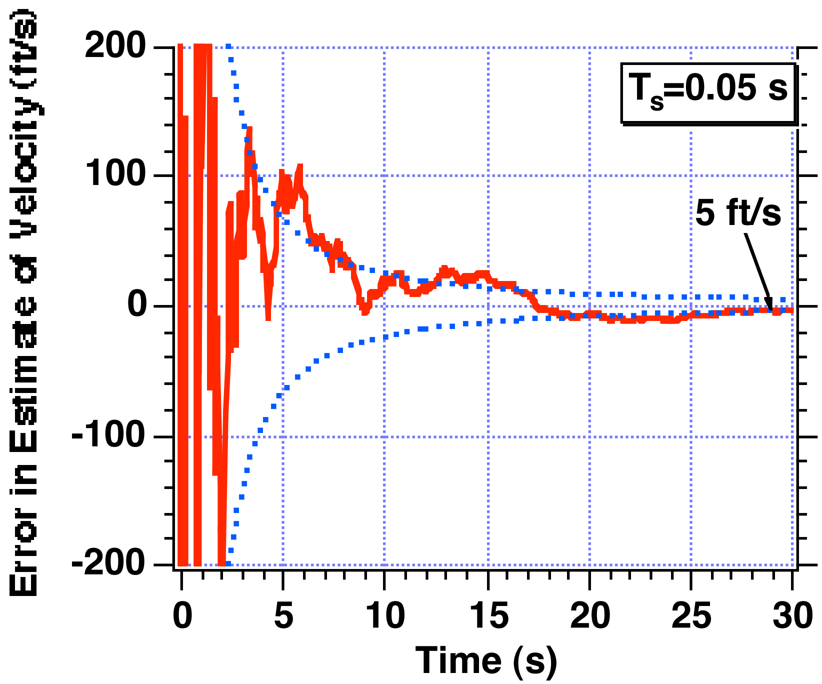
Average sampling time is 0.05 s

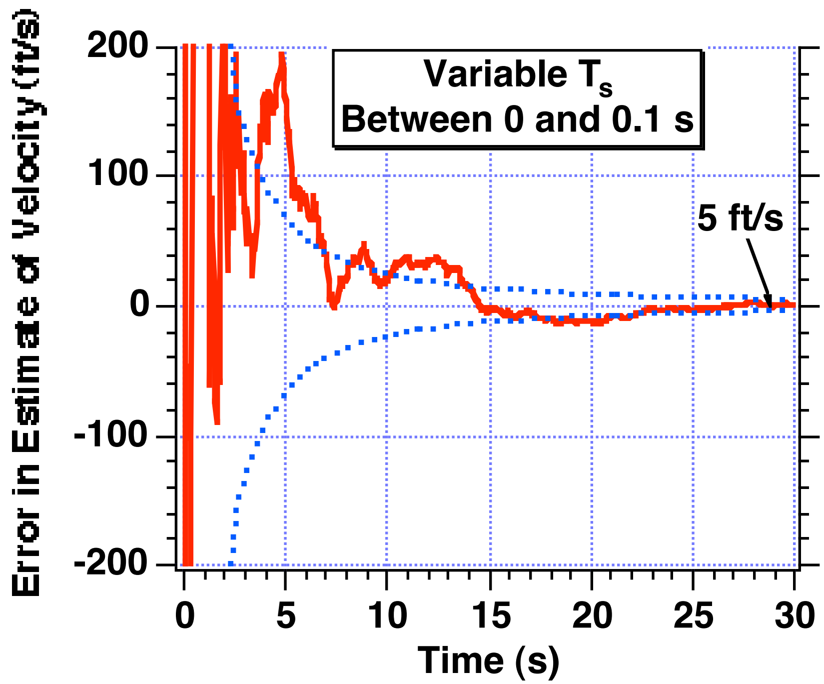
Variable Sampling Time Filter Gives Good Position Estimates





Variable Sampling Time Filter Gives Good Velocity Estimates



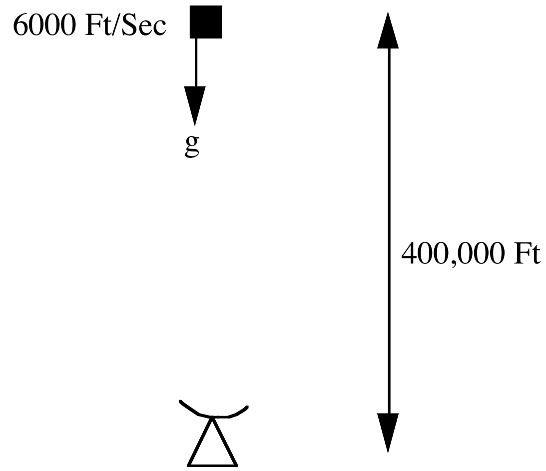


Batch Processing

Batch Processing
Overview

- Problem review
  - Throwing away data
- Preprocessing
  - Wrong way
  - Right way

Radar Tracking Falling Object



From basic physics

$$x = 400000 - 6000t - .5gt^2$$ ← **Second-order process**

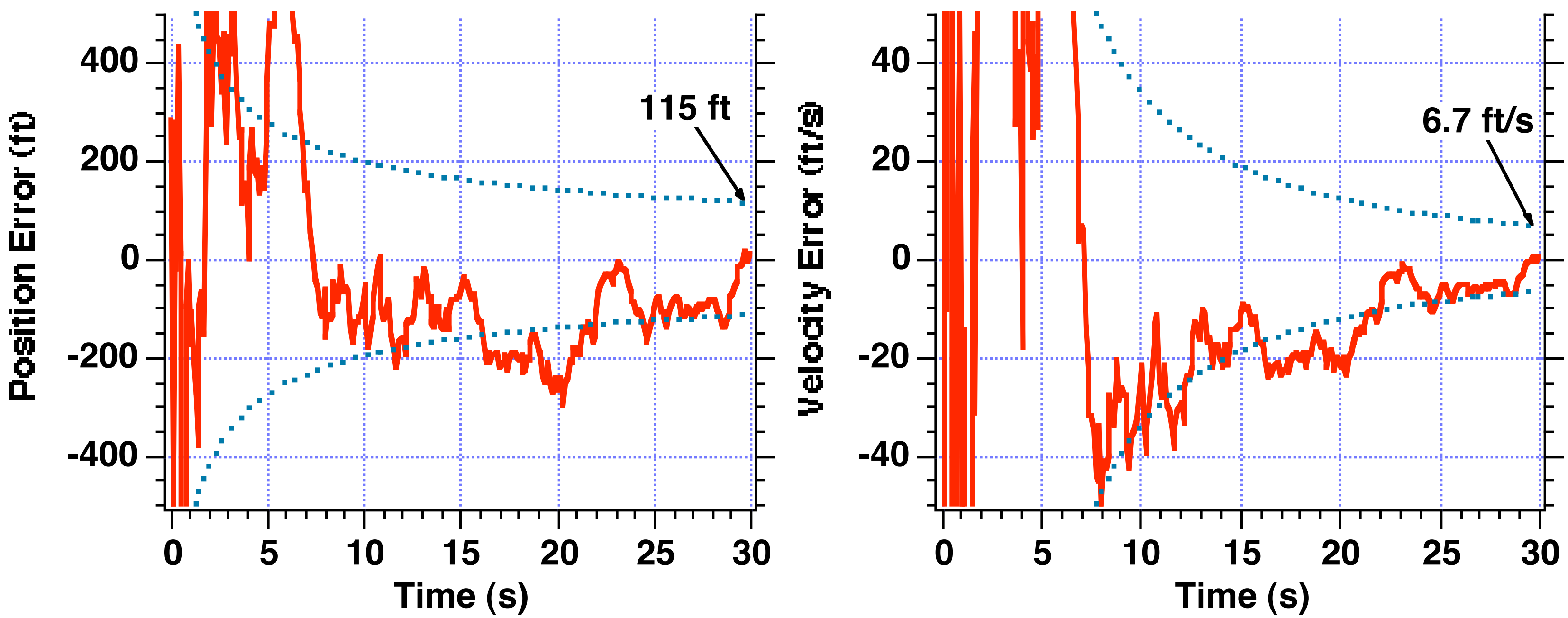
Velocity of object can be found by differentiating

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

Radar measures altitude with standard deviation of 1000 ft

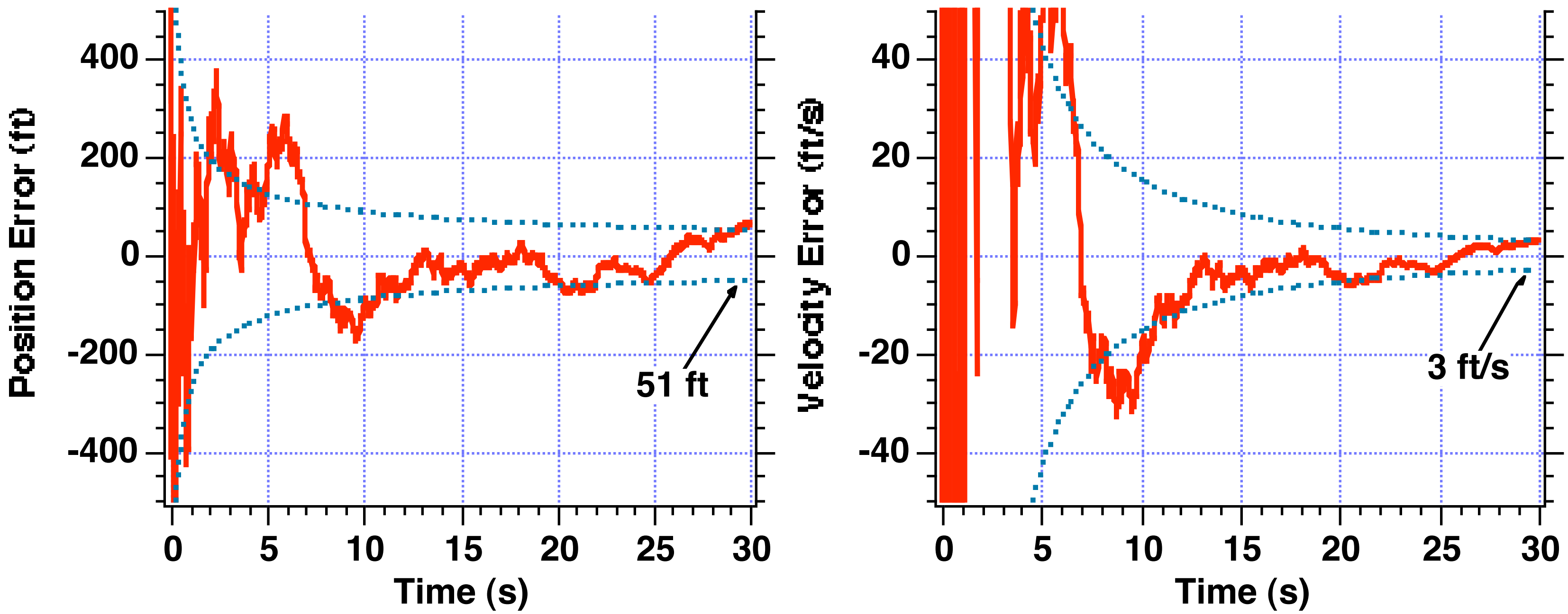
Desire to track object and estimate altitude and velocity

Simple Filtering Every 0.1 s Works



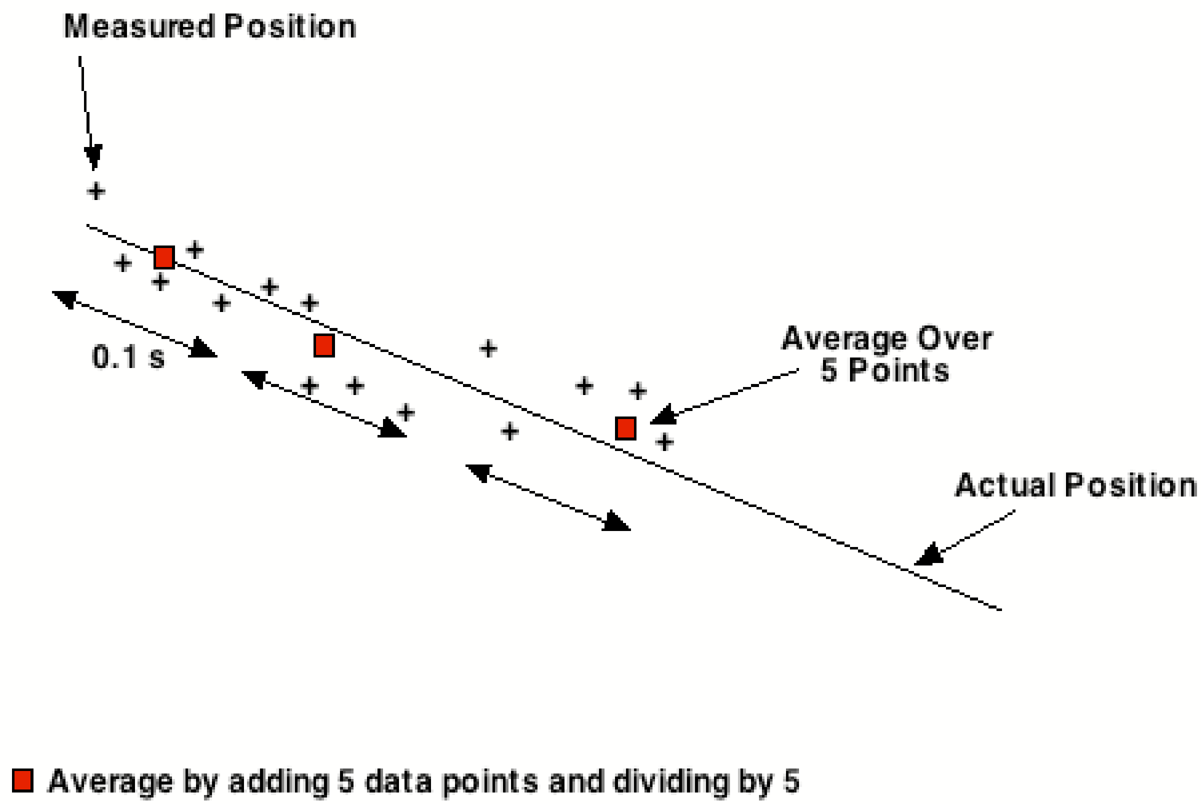
Data available every 0.02 s, but sent to KF every 0.1 s

Simple Filtering Every 0.02 s Works Even Better



Data available every 0.02 s, and sent to KF every 0.02 s

Preprocessing Measurements By Averaging



FORTRAN Code For Batch Processing Scheme-1

```
        GLOBAL DEFINE
                INCLUDE 'quickdraw.inc'
        END
        IMPLICIT REAL*8 (A-H)
        IMPLICIT REAL*8 (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')
        OPEN(2,STATUS='UNKNOWN',FILE='COVFIL')
        PHIS=0.
        TS1=.02
        TS2=.1
        TF=30.
        A0=400000.
        A1=-6000.
        A2=-16.1
        XH=0.
        XDH=0.
        SIGNOISE=1000.
        ORDER=2
        T=0.
        S1=0.
        S2=0.
        H=.001
        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
        IDN(1,1)=1.
        IDN(2,2)=1.
        P(1,1)=99999999999.
        P(2,2)=99999999999.
```

FORTRAN Code For Batch Processing Scheme-2

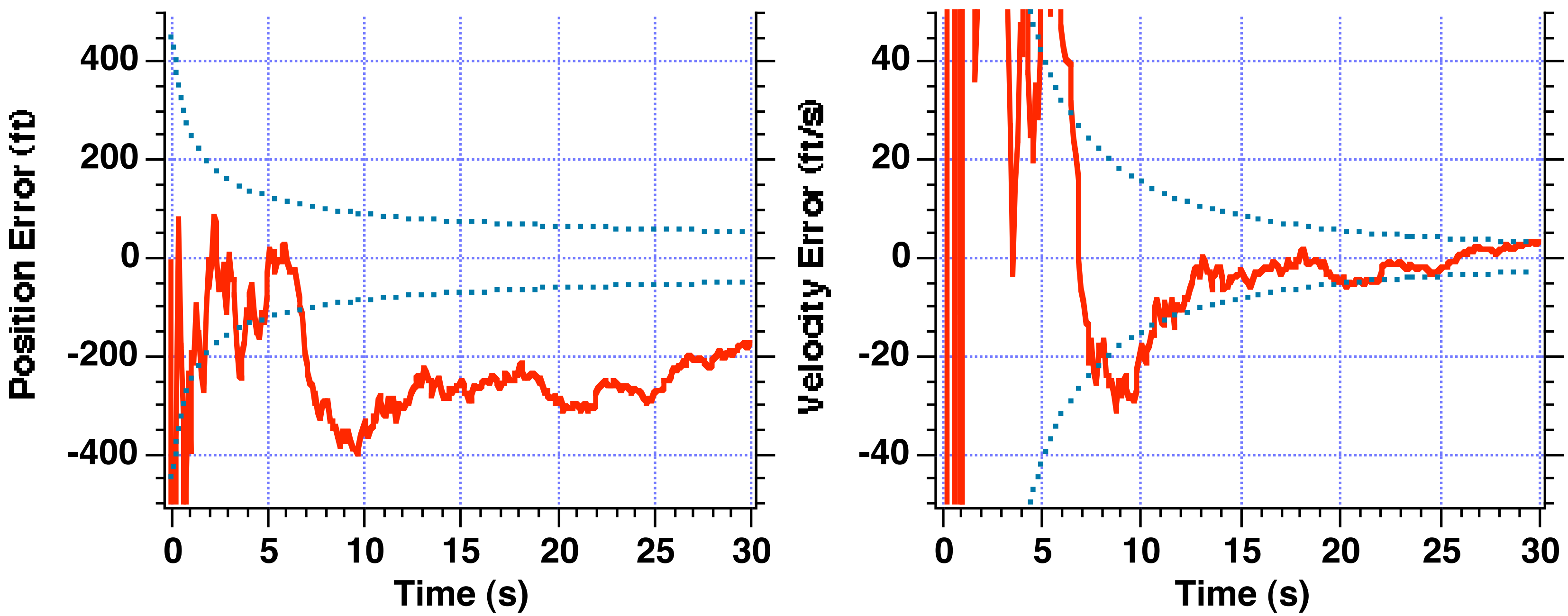
```
PHI(1,1)=1.
PHI(1,2)=TS2
PHI(2,2)=1.
Q(1,1)=TS2*TS2*TS2*PHIS/3.
Q(1,2)=.5*TS2*TS2*PHIS
Q(2,1)=Q(1,2)
Q(2,2)=PHIS*TS2
HMAT(1,1)=1.
HMAT(1,2)=0.
E1=0.
E2=0.
E3=0.
E4=0.
DO 10 T=0.,TF,TS1
        X=A0+A1*T+A2*T*T
        XD=A1+2.*A2*T
        CALL GAUSS(XNOISE,SIGNOISE)
        XS=X+XNOISE
        E=XS
        EAV=(E4+E3+E2+E1+E)/5.
        E4=E3
        E3=E2
        E2=E1
        E1=E
        S2=S2+TS1
```

Average 5 measurements

FORTRAN Code For Batch Processing Scheme-3

```
               IF(S2>=(TS2-.00001))THEN
                    S2=0.
                    RMAT(1,1)=(SIGNOISE/2.24)**2
                    CALL MATTRN(PHI,ORDER,ORDER,PHIT)
                    CALL MATTRN(HMAT,1,ORDER,HT)
                    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)
C                   WRONG WAY
C                   XS=EAV
C                   CORRECT WAY
                    XS=EAV+.05*XDH-16.1*.05*.05
                    RES=XS-XH-TS2*XDH+16.1*TS2*TS2
                    XH=XH+XDH*TS2-16.1*TS2*TS2+K(1,1)*RES
                    XDH=XDH-32.2*TS2+K(2,1)*RES
                    SP11=SQRT(P(1,1))
                    SP22=SQRT(P(2,2))
                    XHERR=X-XH
                    XDHERR=XD-XDH
                    WRITE(9,*)T,X,XH,XD,XDH
                    WRITE(1,*)T,X,XH,XD,XDH
                    WRITE(2,*)T,XHERR,SP11,-SP11,XDHERR,SP22,-SP22
               ENDIF
 10       CONTINUE
          PAUSE
          END
```

Batch Processing Yields Large Hangoff Error in Position When Measurement is Just Averaged

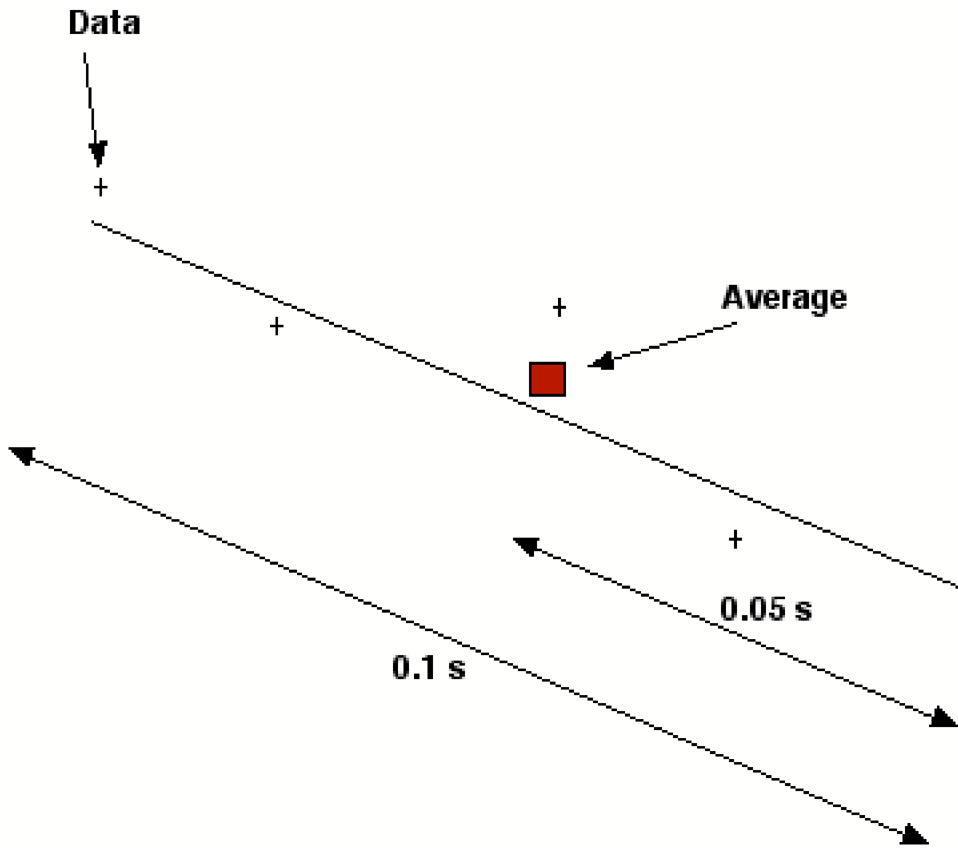


FORTRAN

XS=EAV

Data available every 0.02 s, averaged and sent to KF every 0.1 s

We Must Propagate Average Forward In Order To Be Used as a Measurement

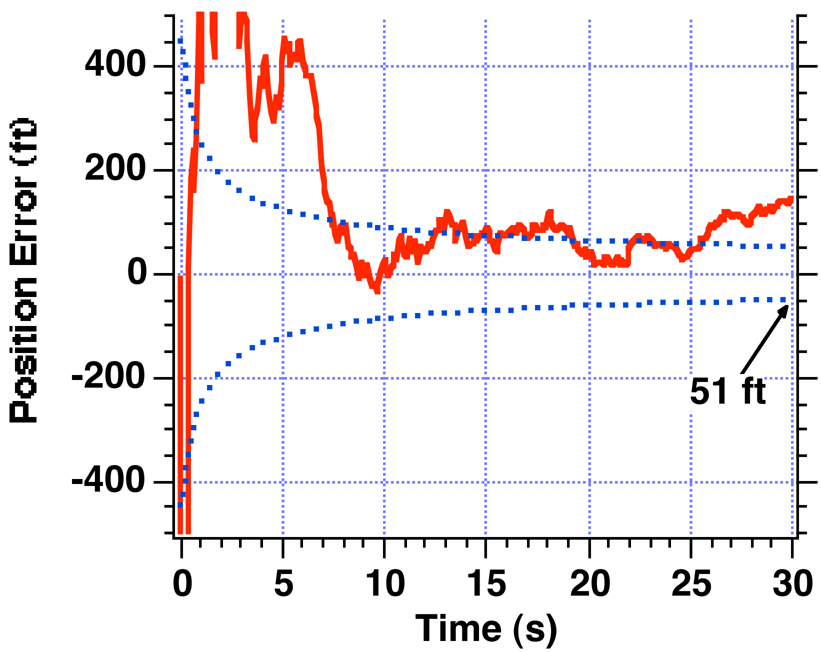


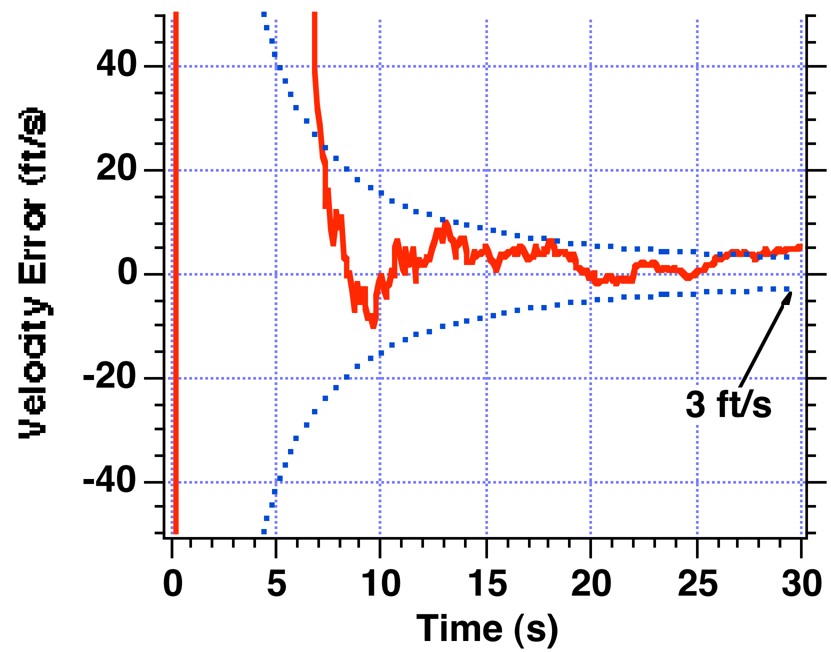
Measurement = Average + .05 $\hat{\dot{x}}$ -16.1*.05*.05

FORTRAN

```
XS=EAV+.05*XDH-16,1*.05*.05
```

Hangoff Error is Mostly Removed and 0.02 s Performance Achieved When Measurement is Averaged and Propagated





FORTRAN

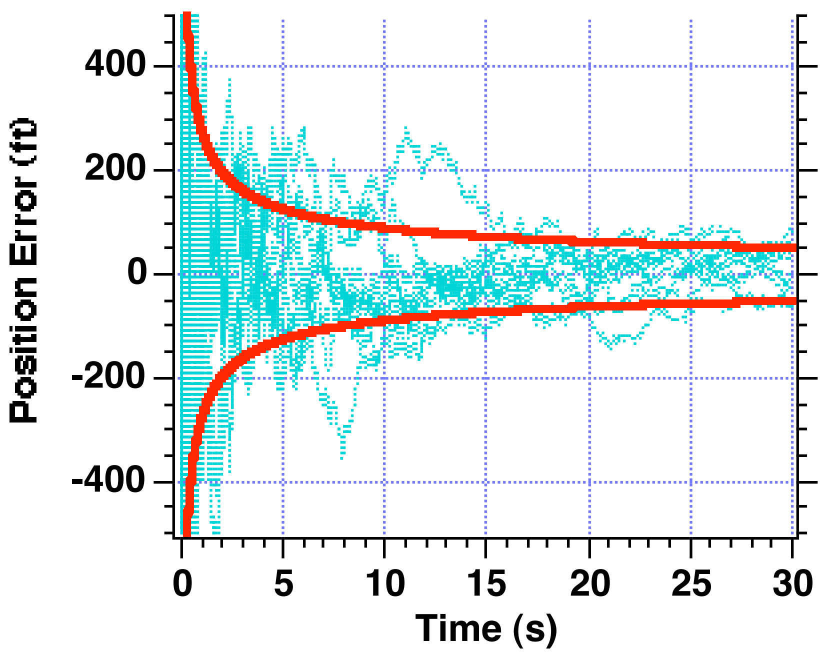
XS=EAV+.05*XDH-16,1*.05*.05

Data available every 0.02 s, averaged and sent to KF every 0.1 s

Monte Carlo Position Error (10 Runs)

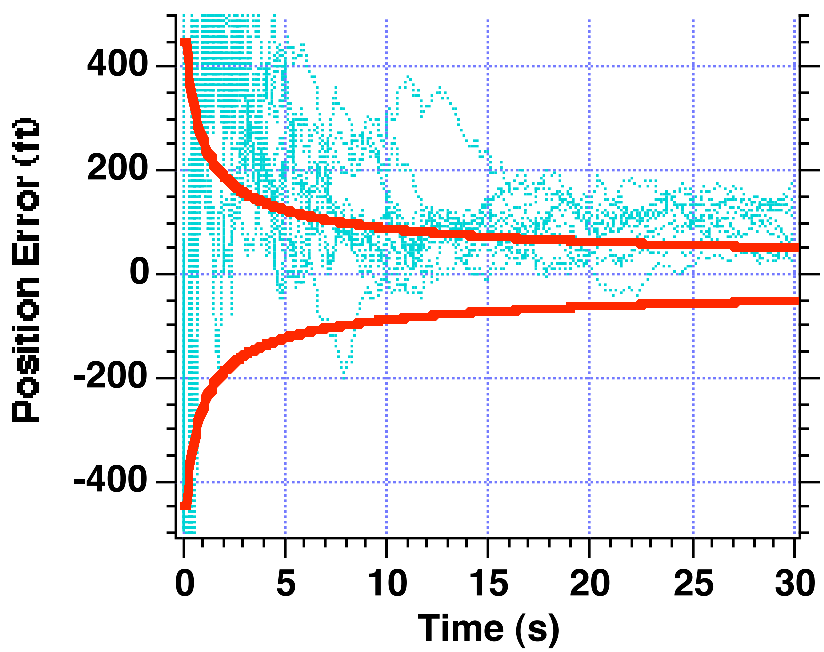
Data available every 0.02 s

Sent to KF every 0.02 s



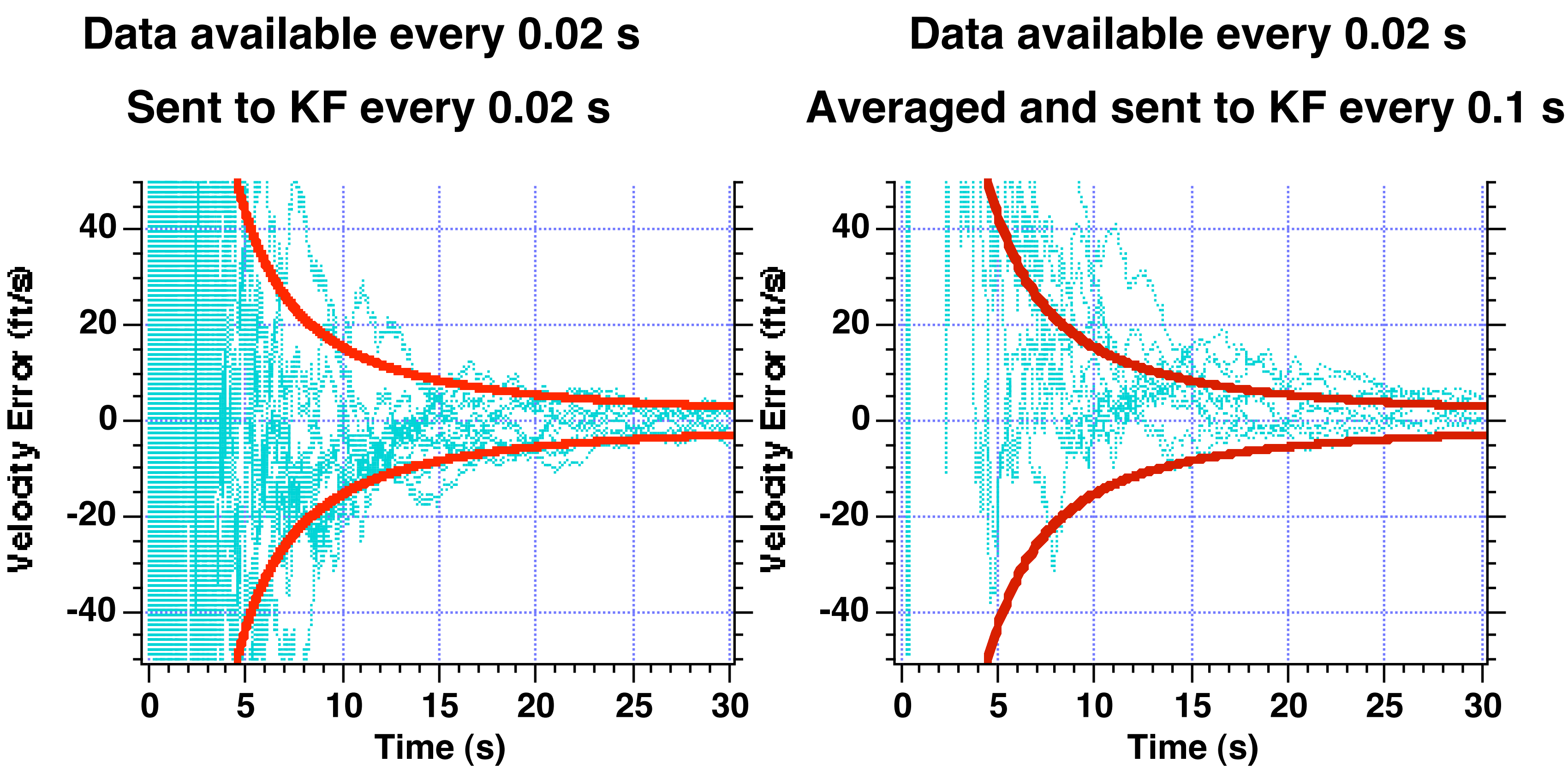
Data available every 0.02 s

Averaged and sent to KF every 0.1 s

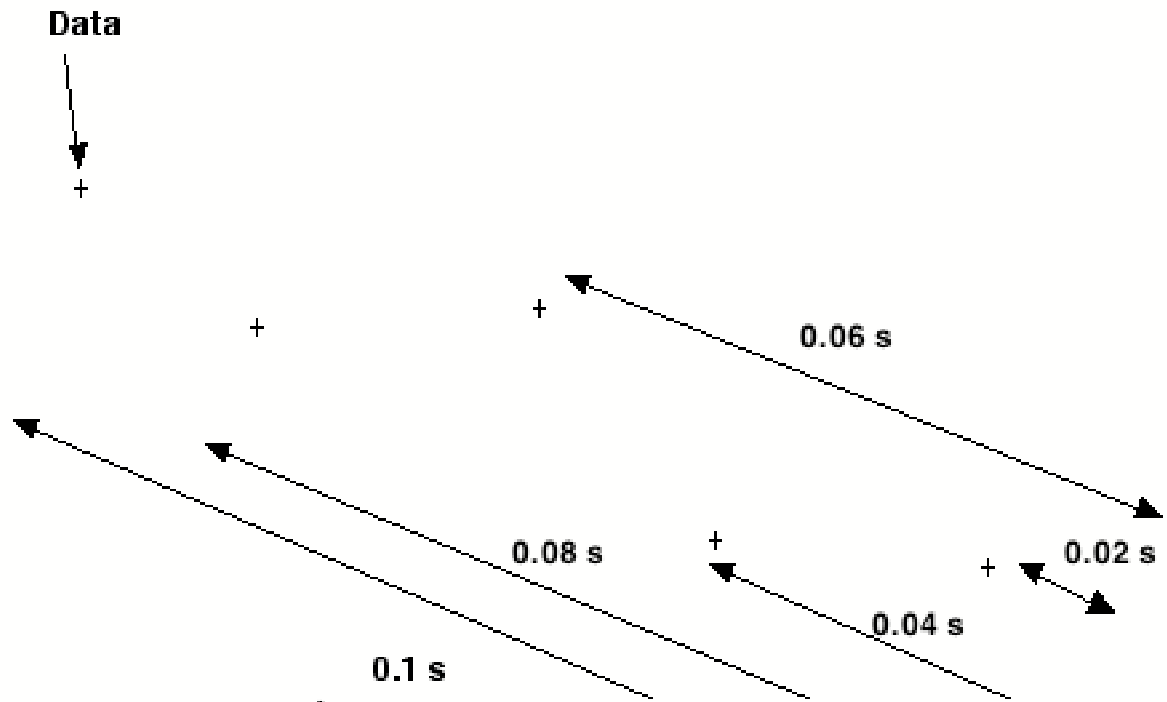


There is still a slight bias in position error when measurement has been averaged and propagated

Monte Carlo Velocity Error (10 Runs)



There is virtually no bias in velocity error when measurement has been averaged and propagated



$$\text{Proj1} = \text{Measurement} + 0.1\hat{\dot{x}} - 16.1*0.1*0.1$$

$$\text{Proj2} = \text{Measurement} + 0.08\hat{\dot{x}} - 16.1*0.08*0.08$$

$$\text{Proj3} = \text{Measurement} + 0.06\hat{\dot{x}} - 16.1*0.06*0.06$$

$$\text{Proj4} = \text{Measurement} + 0.04\hat{\dot{x}} - 16.1*0.04*0.04$$

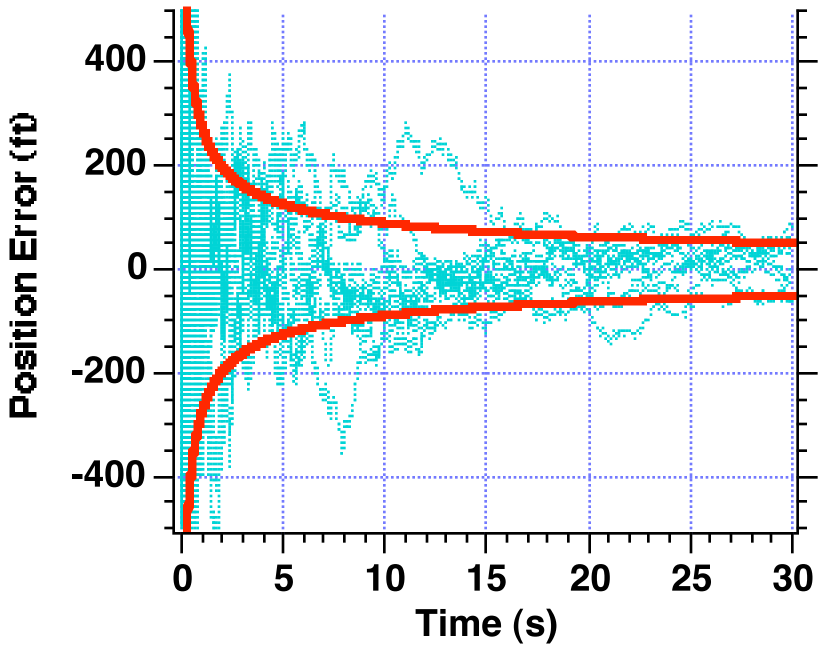
$$\text{Proj5} = \text{Measurement} + 0.02\hat{\dot{x}} - 16.1*0.02*0.02$$

$$\text{Average Measurement} \quad \frac{\text{Proj1+Proj2+Proj3+Proj4+Pro}}{5}$$

New Monte Carlo Position Error (10 Runs)

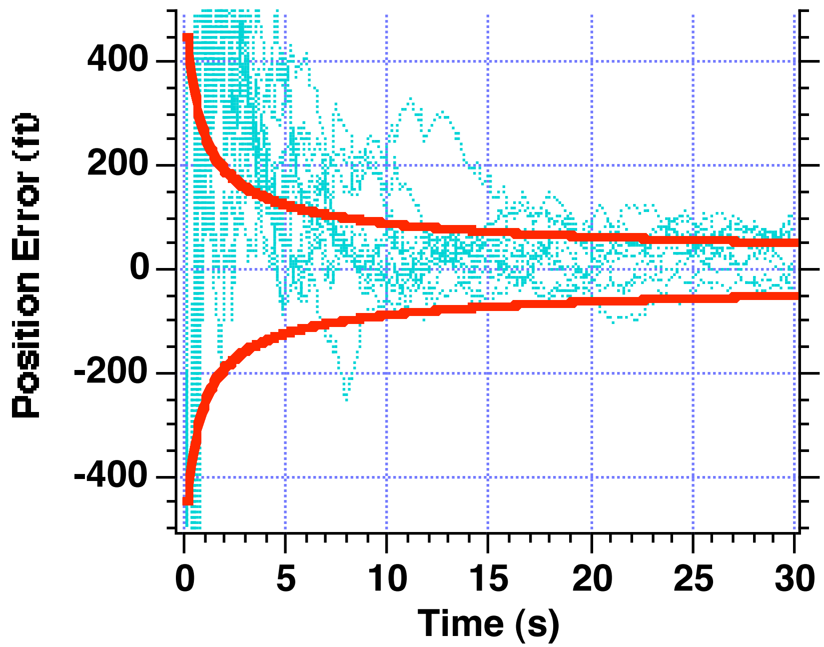
Data available every 0.02 s

Sent to KF every 0.02 s



Data available every 0.02 s

Averaged and sent to KF every 0.1 s

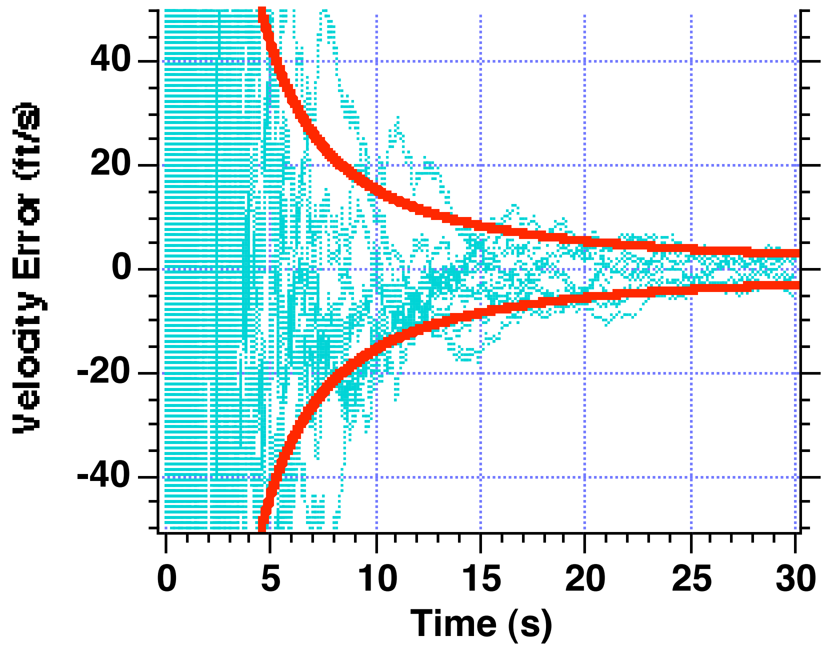


Now there is no bias in position error when measurement has been propagated and averaged

New Monte Carlo Velocity Error (10 Runs)

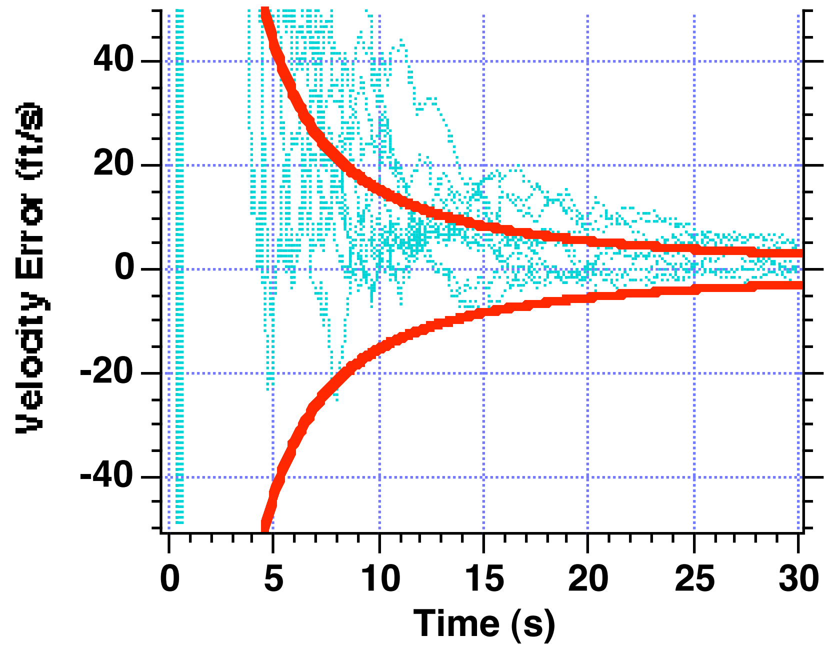
Data available every 0.02 s

Sent to KF every 0.02 s



Data available every 0.02 s

Averaged and sent to KF every 0.1 s



There is still virtually no bias in velocity error when measurement has been propagated and averaged