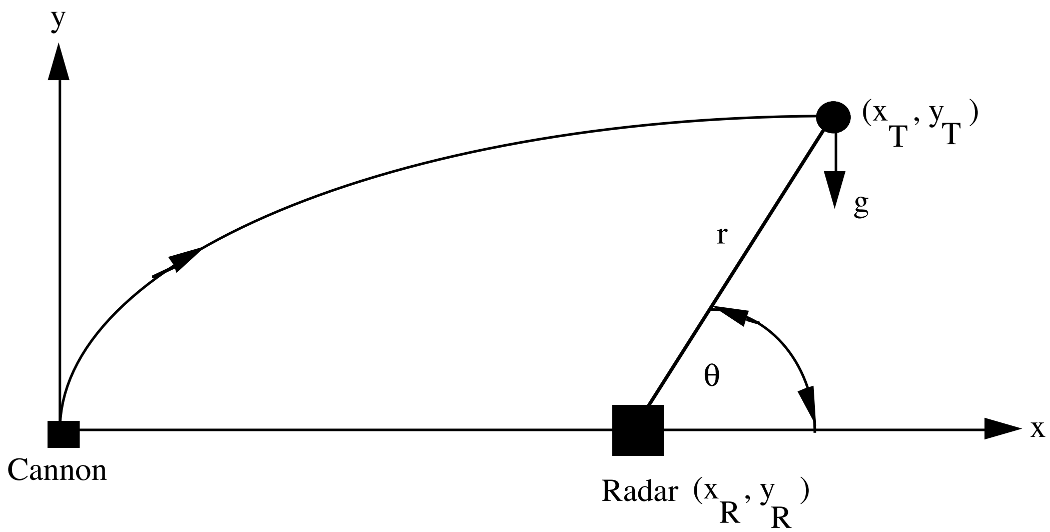
Cannon Launched Projectile Tracking Problem

Cannon Launched Projectile Tracking Problem Overview

- Problem viewed in Cartesian coordinates
- Extended Cartesian Kalman filter
- Problem viewed in polar coordinates
- Extended polar Kalman filter
- Using linear decoupled polynomial Kalman filters
- Using linear coupled polynomial Kalman filters
- Robustness comparison of extended and linear coupled polynomial Kalman filters

Problem Viewed in Cartesian Coordinates

Radar Tracking Cannon Launched Projectile



Relevant Equations

Acceleration of projectile

$$\ddot{x}_T = 0$$

$$\ddot{y}_T = -g$$

Range and angle from radar to projectile

$$\theta = \tan^{-1}\left[\frac{y_T - y_R}{x_T - x_R}\right]$$

$$r = \sqrt{(x_T - x_R)^2 + (y_T - y_R)^2}$$

Actual location of projectile in terms of radar parameters

$$x_T = r\cos\theta + x_R$$

$$y_T = r\sin\theta + y_R$$

Using Raw Radar Measurements to Estimate Projectile Position and Velocity

Recall

$$x_T = r\cos\theta + x_R$$

$$y_T = r\sin\theta + y_R$$

Estimated location of projectile in terms of raw radar measurements

$$\hat{x}_T = r^*\cos\theta^* + x_R$$

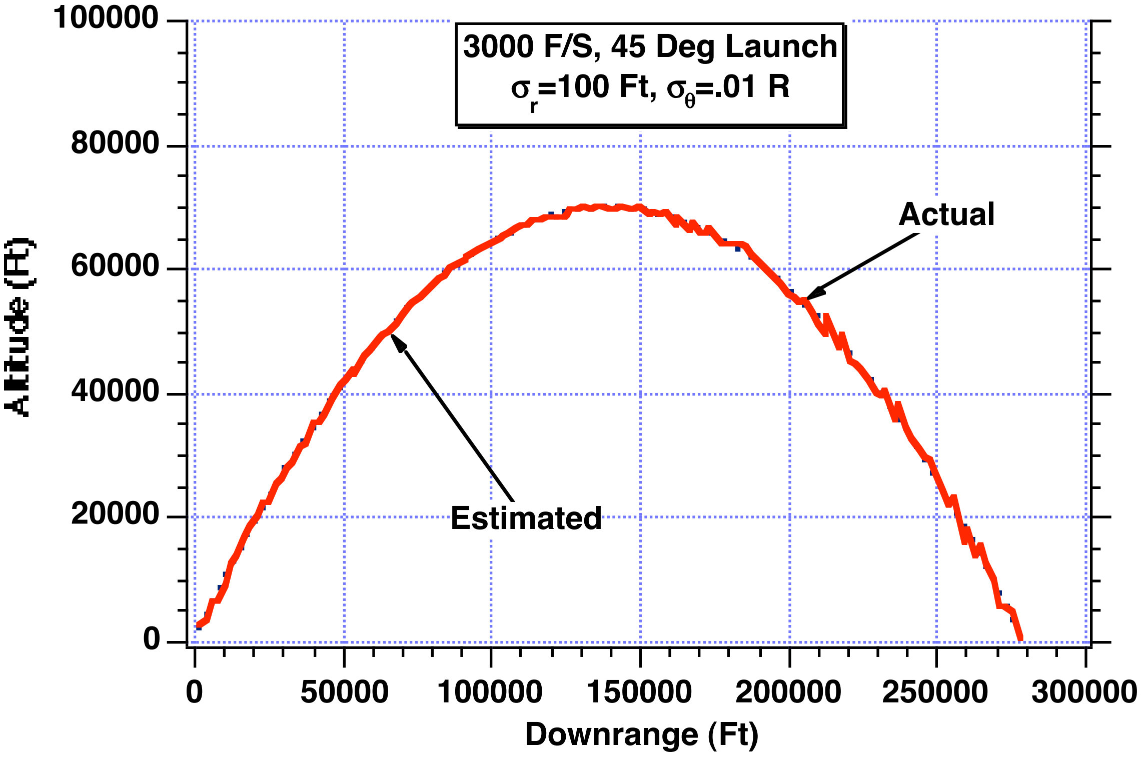
$$\hat{y}_T = r^*\sin\theta^* + y_R$$

Estimated velocity of projectile in terms of raw radar measurements

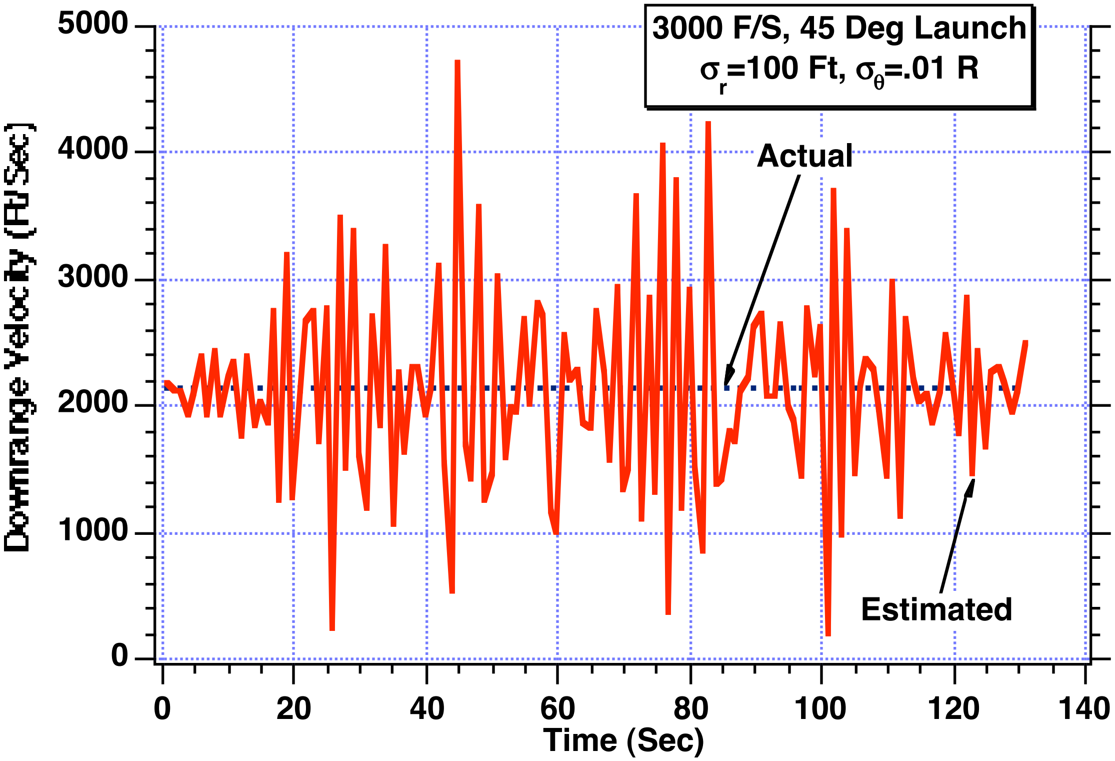
$$\hat{\dot{x}}_{T_k} = \frac{\hat{x}_{T_k} - \hat{x}_{T_{k-1}}}{T_s}$$

$$\hat{\dot{y}}_{T_k} = \frac{\hat{y}_{T_k} - \hat{y}_{T_{k-1}}}{T_s}$$

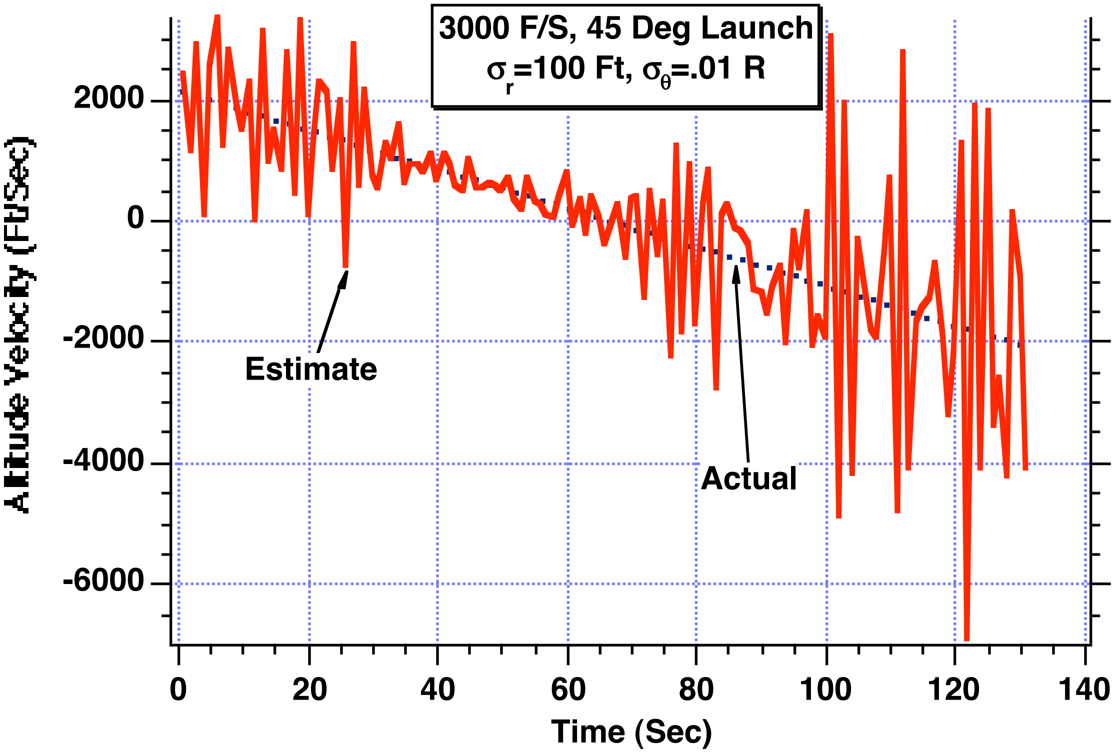
Using Raw Measurements to Estimate Trajectory Appears to be Satisfactory



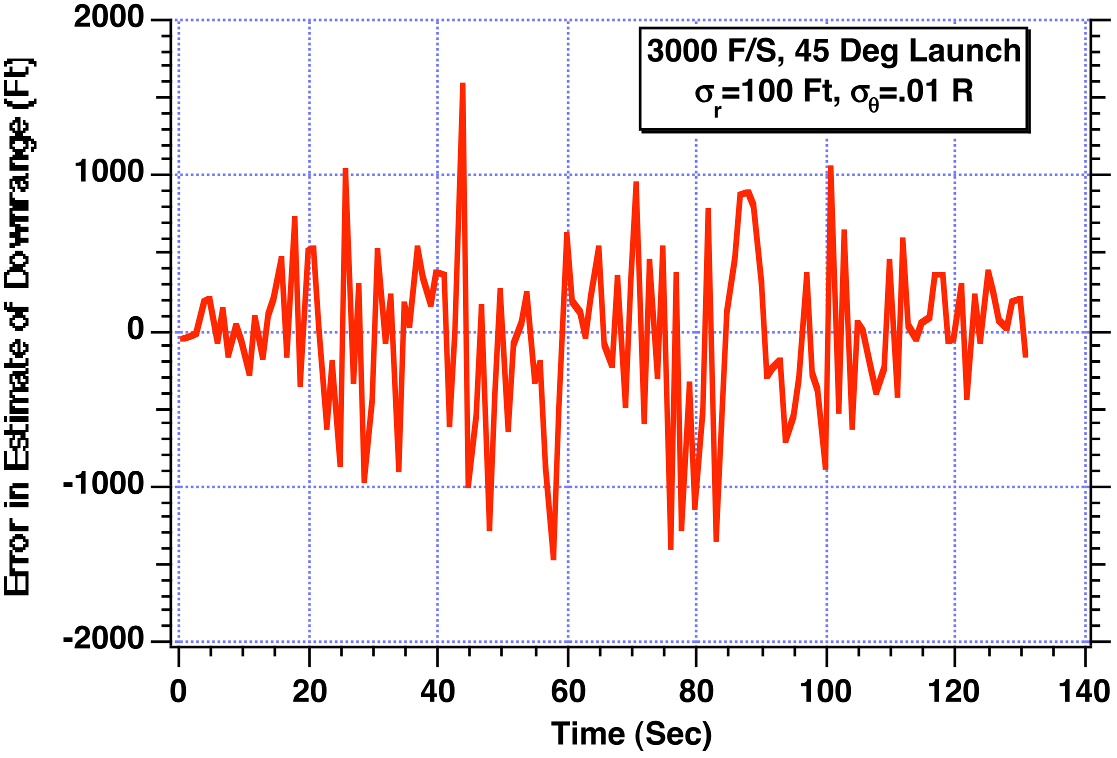
Using Raw Measurements Yields Terrible Downrange Velocity Estimates



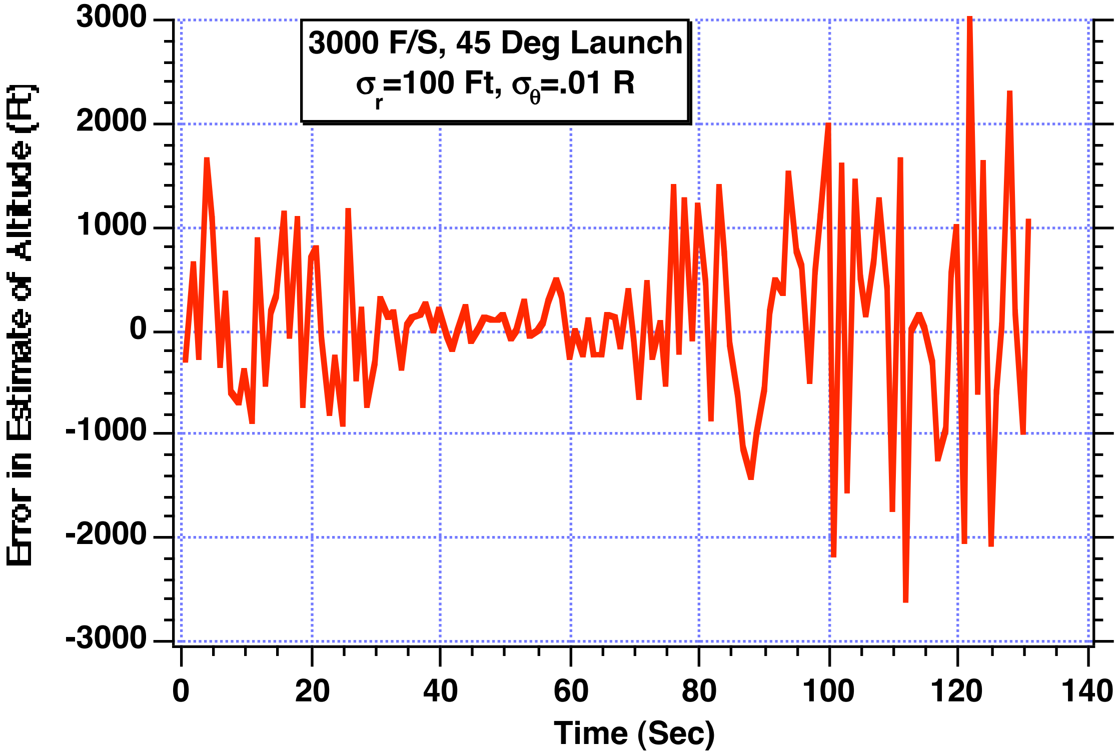
Using Raw Measurements Also Yields Very Poor Altitude Velocity Estimates



Error in Estimate of Downrange is Often Greater Than 1000 Ft



Error in Estimate of Altitude is Often Greater Than 1000 Ft



Extended Cartesian Kalman Filter

Setting up extended Kalman filter

Choice of states

$$\mathbf{x} = \begin{bmatrix} x_T \\ \dot{x}_T \\ y_T \\ \dot{y}_T \end{bmatrix}$$

Model of real world

$$\begin{bmatrix} \dot{x}_T \\ \ddot{x}_T \\ \dot{y}_T \\ \ddot{y}_T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_T \\ \dot{x}_T \\ y_T \\ \dot{y}_T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} + \left[\begin{bmatrix} 0 \\ u_s \\ 0 \\ u_s \end{bmatrix} \right]$$

Systems dynamics matrix

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

Continuous process noise matrix

$$\mathbf{Q}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Phi_s & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Phi_s \end{bmatrix}$$

Deriving Fundamental Matrix

Recall

$$\mathbf{\Phi}(t) = \mathbf{I} + \mathbf{F}t + \frac{\mathbf{F}^2t^2}{2!} + \frac{\mathbf{F}^3t^3}{3!} + \ldots$$

Since

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

We get

$$\mathbf{\Phi}(t) = \mathbf{I} + \mathbf{F}t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} t \longrightarrow \mathbf{\Phi}(t) = \begin{bmatrix} 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Discrete fundamental matrix

$$\mathbf{\Phi}_\mathbf{k} = \begin{bmatrix} 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Developing Measurement Matrix-1

Measurement equation is nonlinear

$$\theta = \tan^{-1}\left[\frac{y_T - y_R}{x_T - x_R}\right]$$

Linearized measurement equation

$$\begin{bmatrix} \Delta\theta^* \\ \Delta r^* \end{bmatrix} = \begin{bmatrix} \frac{\partial\theta}{\partial x_T} & \frac{\partial\theta}{\partial \dot{x}_T} & \frac{\partial\theta}{\partial y_T} & \frac{\partial\theta}{\partial \dot{y}_T} \\ \frac{\partial r}{\partial x_T} & \frac{\partial r}{\partial \dot{x}_T} & \frac{\partial r}{\partial y_T} & \frac{\partial r}{\partial \dot{y}_T} \end{bmatrix} \begin{bmatrix} \Delta x_T \\ \Delta \dot{x}_T \\ \Delta y_T \\ \Delta \dot{y}_T \end{bmatrix} + \begin{bmatrix} v_\theta \\ v_r \end{bmatrix}$$

First row can be evaluated as

$$\frac{\partial\theta}{\partial x_T} = \frac{1}{1+\frac{(y_T - y_R)^2}{(x_T - x_R)^2}} \frac{(x_T - x_R)*0 - (y_T - y_R)*1}{(x_T - x_R)^2} = \frac{-(y_T - y_R)}{r^2}$$

$$\frac{\partial\theta}{\partial \dot{x}_T} = 0$$

$$\frac{\partial\theta}{\partial y_T} = \frac{1}{1+\frac{(y_T - y_R)^2}{(x_T - x_R)^2}} \frac{(x_T - x_R)*1 - (y_T - y_R)*0}{(x_T - x_R)^2} = \frac{(x_T - x_R)}{r^2}$$

$$\frac{\partial\theta}{\partial \dot{y}_T} = 0$$

Developing Measurement Matrix-2

Since

$$r = \sqrt{(x_T - x_R)^2 + (y_T - y_R)^2}$$

Second row can be evaluated as

$$\frac{\partial r}{\partial x_T} = \frac{1}{2}\,[(x_T - x_R)^2 + (y_T - y_R)^2]^{-1/2}\,2(x_T - x_R) = \frac{(x_T - x_R)}{r}$$

$$\frac{\partial r}{\partial \dot{x}_T} = 0$$

$$\frac{\partial r}{\partial y_T} = \frac{1}{2}\,[(x_T - x_R)^2 + (y_T - y_R)^2]^{-1/2}\,2(y_T - y_R) = \frac{(y_T - y_R)}{r}$$

$$\frac{\partial r}{\partial \dot{y}_T} = 0$$

Measurement matrix

$$\begin{tikzcd}
\mathbf{H} = \begin{bmatrix} \frac{\partial \theta}{\partial x_T} & \frac{\partial \theta}{\partial \dot{x}_T} & \frac{\partial \theta}{\partial y_T} & \frac{\partial \theta}{\partial \dot{y}_T} \\ \frac{\partial r}{\partial x_T} & \frac{\partial r}{\partial \dot{x}_T} & \frac{\partial r}{\partial y_T} & \frac{\partial r}{\partial \dot{y}_T} \end{bmatrix} \arrow[r] & \mathbf{H} = \begin{bmatrix} \frac{-(y_T - y_R)}{r^2} & 0 & \frac{x_T - x_R}{r^2} & 0 \\ \frac{x_T - x_R}{r} & 0 & \frac{y_T - y_R}{r} & 0 \end{bmatrix} \arrow[d] \\
& \mathbf{H_k} = \begin{bmatrix} \frac{-(\bar{y}_{T_k} - y_R)}{\bar{r}_k^2} & 0 & \frac{\bar{x}_{T_k} - x_R}{\bar{r}_k^2} & 0 \\ \frac{\bar{x}_{T_k} - x_R}{\bar{r}_k} & 0 & \frac{\bar{y}_{T_k} - y_R}{\bar{r}_k} & 0 \end{bmatrix}
\end{tikzcd}$$

Other Important Matrices

Measurement noise matrix

$$\mathbf{R_k} = E(\mathbf{v_k v_k^T}) \longrightarrow \mathbf{R_k} = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

Discrete process noise matrix

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

$$\mathbf{Q_k} = \int_0^{T_s} \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Phi_s & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Phi_s \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ \tau & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \tau & 1 \end{bmatrix} d\tau$$

$$\mathbf{Q_k} = \int_0^{T_s} \begin{bmatrix} \tau^2\Phi_s & \tau\Phi_s & 0 & 0 \\ \tau\Phi_s & \Phi_s & 0 & 0 \\ 0 & 0 & \tau^2\Phi_s & \tau\Phi_s \\ 0 & 0 & \tau\Phi_s & \Phi_s \end{bmatrix} d\tau \longrightarrow \mathbf{Q_k} = \begin{bmatrix} \frac{T_s^3\Phi_s}{3} & \frac{T_s^2\Phi_s}{2} & 0 & 0 \\ \frac{T_s^2\Phi_s}{2} & T_s\Phi_s & 0 & 0 \\ 0 & 0 & \frac{T_s^3\Phi_s}{3} & \frac{T_s^2\Phi_s}{2} \\ 0 & 0 & \frac{T_s^2\Phi_s}{2} & T_s\Phi_s \end{bmatrix}$$

Discrete G Matrix

Filter equation

$$\dot{x} = Fx + Gu + w \longrightarrow \hat{x}_k = \Phi_k \hat{x}_{k-1} + G_k u_{k-1} + K_k(z_k - H\Phi_k \hat{x}_{k-1} - HG_k u_{k-1})$$

If u constant between sampling instants discrete G matrix from

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

Recall

$$\begin{bmatrix} \dot{x}_T \\ \ddot{x}_T \\ \dot{y}_T \\ \ddot{y}_T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_T \\ \dot{x}_T \\ y_T \\ \dot{y}_T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} + \left[\begin{bmatrix} 0 \\ u_s \\ 0 \\ u_s \end{bmatrix}\right] \longrightarrow G = Gu = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix}$$

Therefore

$$G_k = \int_0^{T_s} \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} d\tau = \begin{bmatrix} 0 \\ 0 \\ -\dfrac{gT_s^2}{2} \\ -gT_s \end{bmatrix}$$

Calculating Projected States

Recall

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

Since fundamental matrix exact projected state is

$$\bar{\mathbf{x}}_{\mathbf{k}} = \mathbf{\Phi}_{\mathbf{k}}\hat{\mathbf{x}}_{\mathbf{k}-1} + \mathbf{G}_{\mathbf{k}}\mathbf{u}_{\mathbf{k}-1}$$

Substitution yields

$$\bar{\mathbf{x}}_{\mathbf{k}} = \begin{bmatrix} 1 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix}\hat{\mathbf{x}}_{\mathbf{k}-1} + \begin{bmatrix} 0 \\ 0 \\ -\dfrac{gT_s^2}{2} \\ -gT_s \end{bmatrix}$$

Multiplying out terms yields

$$\bar{x}_{T_k} = \hat{x}_{T_{k-1}} + T_s\hat{\dot{x}}_{T_{k-1}}$$

$$\overline{\dot{x}}_{T_k} = \hat{\dot{x}}_{T_{k-1}}$$

$$\bar{y}_{T_k} = \hat{y}_{T_{k-1}} + T_s\hat{\dot{y}}_{T_{k-1}} - .5gT_s^2$$

$$\overline{\dot{y}}_{T_k} = \hat{\dot{y}}_{T_{k-1}} - gT_s$$

Extended Kalman Filtering Equations

Recall

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

Nonlinear

Calculate projected measurements

$$\bar{\theta}_k = \tan^{-1}\left[\frac{\bar{y}_{T_{k-1}} - y_R}{\bar{x}_{T_{k-1}} - x_R}\right]$$

$$\bar{r}_k = \sqrt{(\bar{x}_{T_{k-1}} - x_R)^2 + (\bar{y}_{T_{k-1}} - y_R)^2}$$

Filtering equations

$$\hat{x}_{T_k} = \bar{x}_{T_k} + K_{11_k}(\theta_k^* - \bar{\theta}_k) + K_{12_k}(r_k^* - \bar{r}_k)$$

$$\hat{\dot{x}}_{T_k} = \bar{\dot{x}}_{T_k} + K_{21_k}(\theta_k^* - \bar{\theta}_k) + K_{22_k}(r_k^* - \bar{r}_k)$$

$$\hat{y}_{T_k} = \bar{y}_{T_k} + K_{31_k}(\theta_k^* - \bar{\theta}_k) + K_{32_k}(r_k^* - \bar{r}_k)$$

$$\hat{\dot{y}}_{T_k} = \bar{\dot{y}}_{T_k} + K_{41_k}(\theta_k^* - \bar{\theta}_k) + K_{42_k}(r_k^* - \bar{r}_k)$$

Where

$$\bar{x}_{T_k} = \hat{x}_{T_{k-1}} + T_s\hat{\dot{x}}_{T_{k-1}}$$

$$\bar{\dot{x}}_{T_k} = \hat{\dot{x}}_{T_{k-1}}$$

$$\bar{y}_{T_k} = \hat{y}_{T_{k-1}} + T_s\hat{\dot{y}}_{T_{k-1}} -.5gT_s^2$$

$$\bar{\dot{y}}_{T_k} = \hat{\dot{y}}_{T_{k-1}} - gT_s$$

Note that linearized measurement is not use here but is only used in the Riccati equation

MATLAB Cartesian Extended Kalman Filter-1

```
TS=1.;
ORDER=4;
PHIS=0.;
SIGTH=.01;
SIGR=100.;
VT=3000.;
GAMDEG=45.;
G=32.2;
XT=0.;
YT=0.;
XTD=VT*cos(GAMDEG/57.3);
YTD=VT*sin(GAMDEG/57.3);
```

Actual initial states

```
XR=100000.;
YR=0.;
T=0.;
S=0.;
H=.001;
PHI=[1 TS 0 0;0 1 0 0;0 0 1 TS;0 0 0 1];
P=[1000.^2 0 0 0;0 100.^2 0 0;0 0 1000.^2 0;0 0 0 100.^2];
```

Fundamental and initial covariance matrices

```
IDNP=eye(ORDER);
Q=zeros(ORDER,ORDER);
TS2=TS*TS;
TS3=TS2*TS;
Q(1,1)=PHIS*TS3/3.;
Q(1,2)=PHIS*TS2/2.;
Q(2,1)=Q(1,2);
Q(2,2)=PHIS*TS;
Q(3,3)=Q(1,1);
Q(3,4)=Q(1,2);
Q(4,3)=Q(3,4);
Q(4,4)=Q(2,2);
```

Discrete process noise matrix

```
PHIT=PHI';
RMAT=[SIGTH^2 0;0 SIGR^2];
```

← **Measurement noise matrix**

```
XTH=XT+1000.;
XTDH=XTD-100.;
YTH=YT-1000.;
YTDH=YTD+100.;
```

Initial state estimates

```
count=0;
```

MATLAB Cartesian Extended Kalman Filter-2

```
while YT>=0.
        XTOLD=XT;
        XTDOLD=XTD;
        YTOLD=YT;
        YTDOLD=YTD;
        XTDD=0.;
        YTDD=-G;
        XT=XT+H*XTD;
        XTD=XTD+H*XTDD;
        YT=YT+H*YTD;
        YTD=YTD+H*YTDD;
        T=T+H;
        XTDD=0.;
        YTDD=-G;
        XT=.5*(XTOLD+XT+H*XTD);
        XTD=.5*(XTDOLD+XTD+H*XTDD);
        YT=.5*(YTOLD+YT+H*YTD);
        YTD=.5*(YTDOLD+YTD+H*YTDD);
```

Second-order Runge-Kutta integration of projectile differential equations

```
        S=S+H;
        if S>=(TS-.00001)
                S=0.;
                XTB=XTH+TS*XTDH;
                XTDB=XTDH;
                YTB=YTH+TS*YTDH-.5*G*TS*TS;
                YTDB=YTDH-G*TS;
                RTB=sqrt((XTB-XR)^2+(YTB-YR)^2);
                HMAT(1,1)=-(YTB-YR)/RTB^2;
                HMAT(1,2)=0.;
                HMAT(1,3)=(XTB-XR)/RTB^2;
                HMAT(1,4)=0.;
                HMAT(2,1)=(XTB-XR)/RTB;
                HMAT(2,2)=0.;
                HMAT(2,3)=(YTB-YR)/RTB;
                HMAT(2,4)=0.;
```

Linearized measurement matrix

```
                HT=HMAT';
                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;
```

Riccati equations

MATLAB Cartesian Extended Kalman Filter-3

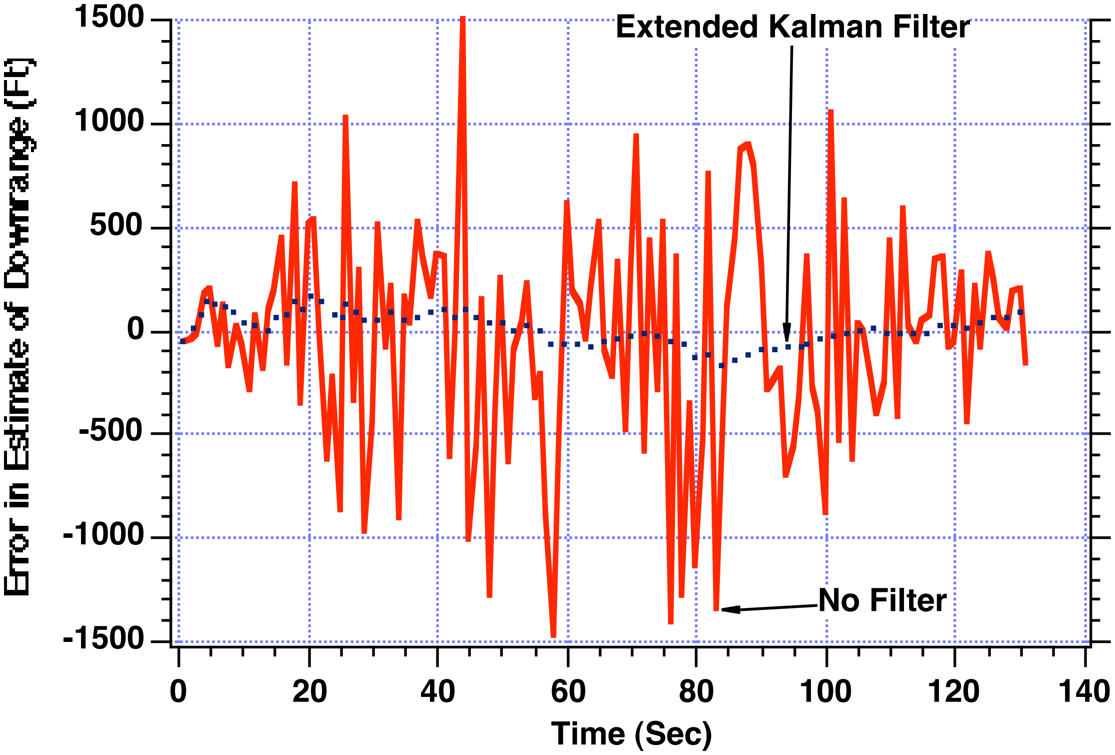
```
                THETNOISE=SIGTH*randn;
                RTNOISE=SIGR*randn;
                THET=atan2((YT-YR),(XT-XR));
                RT=sqrt((XT-XR)^2+(YT-YR)^2);
                THETMEAS=THET+THETNOISE;
                RTMEAS=RT+RTNOISE;
                THETB=atan2((YTB-YR),(XTB-XR));
                RTB=sqrt((XTB-XR)^2+(YTB-YR)^2);
                RES1=THETMEAS-THETB;
                RES2=RTMEAS-RTB;
                XTH=XTB+K(1,1)*RES1+K(1,2)*RES2;
                XTDH=XTDB+K(2,1)*RES1+K(2,2)*RES2;
                YTH=YTB+K(3,1)*RES1+K(3,2)*RES2;
                YTDH=YTDB+K(4,1)*RES1+K(4,2)*RES2;
                ERRX=XT-XTH;
                SP11=sqrt(P(1,1));
                ERRXD=XTD-XTDH;
                SP22=sqrt(P(2,2));
                ERRY=YT-YTH;
                SP33=sqrt(P(3,3));
                ERRYD=YTD-YTDH;
                SP44=sqrt(P(4,4));
                SP11P=-SP11;
                SP22P=-SP22;
                SP33P=-SP33;
                SP44P=-SP44;
                count=count+1;
                ArrayT(count)=T;
                ArrayXT(count)=XT;
                ArrayXTH(count)=XTH;
                ArrayYT(count)=YT;
                ArrayYTH(count)=YTH;
                ArrayXTD(count)=XTD;
                ArrayXTDH(count)=XTDH;
                ArrayYTD(count)=YTD;
                ArrayYTDH(count)=YTDH;
                ArrayERRX(count)=ERRX;
                ArraySP11(count)=SP11;
                ArraySP11P(count)=SP11P;
                ArrayERRXD(count)=ERRXD;
                ArraySP22(count)=SP22;
                ArraySP22P(count)=SP22P;
                ArrayERRY(count)=ERRY;
                ArraySP33(count)=SP33;
                ArraySP33P(count)=SP33P;
                ArrayERRYD(count)=ERRYD;
                ArraySP44(count)=SP44;
                ArraySP44P(count)=SP44P;
        end
end
```

Noisy radar measurements

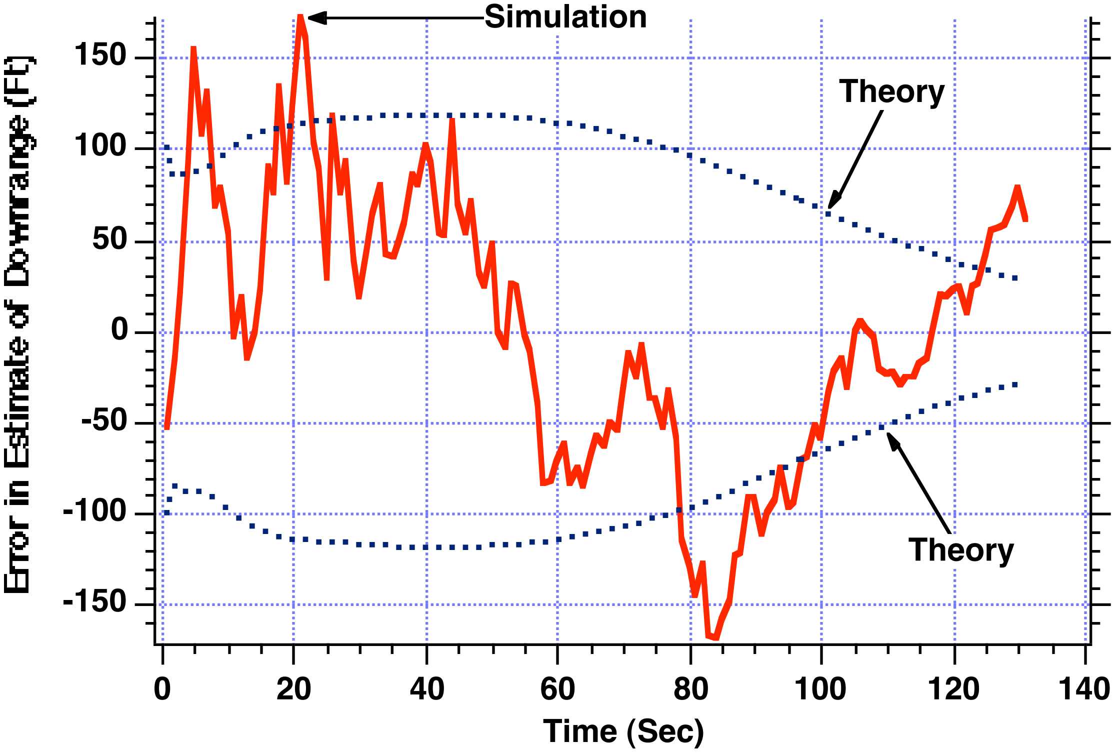
Extended Kalman filter

Saving data as arrays for plotting and writing to files

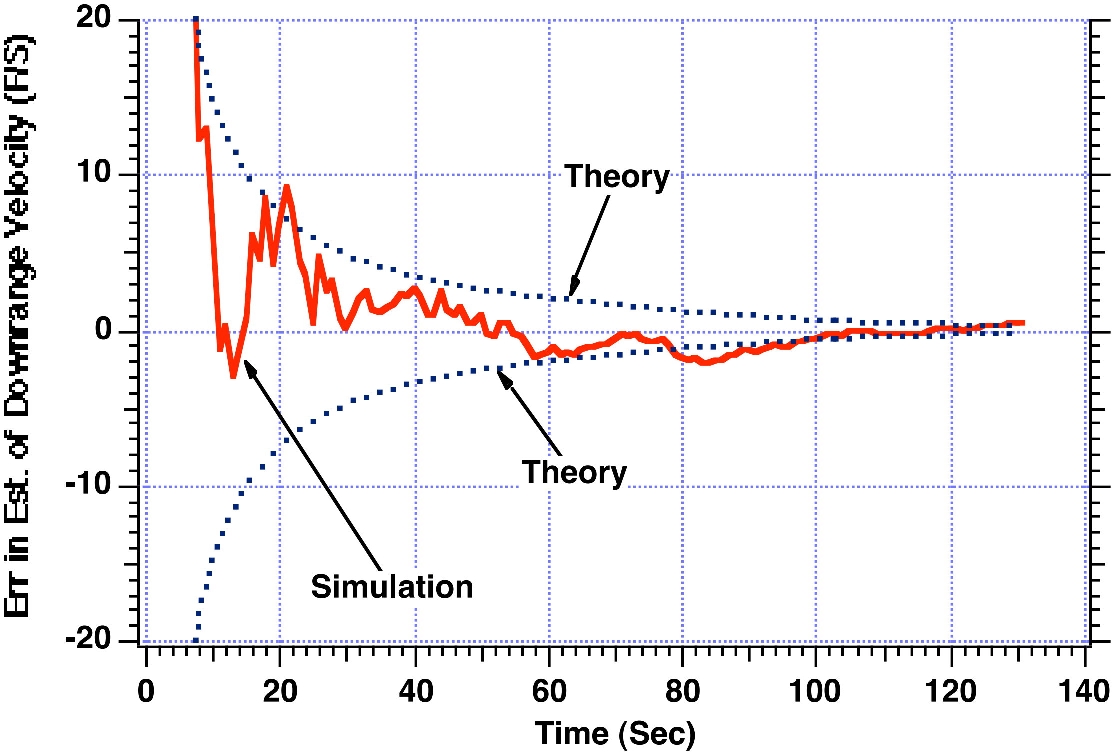
Filtering Dramatically Reduces Position Error Over Using Raw Measurements



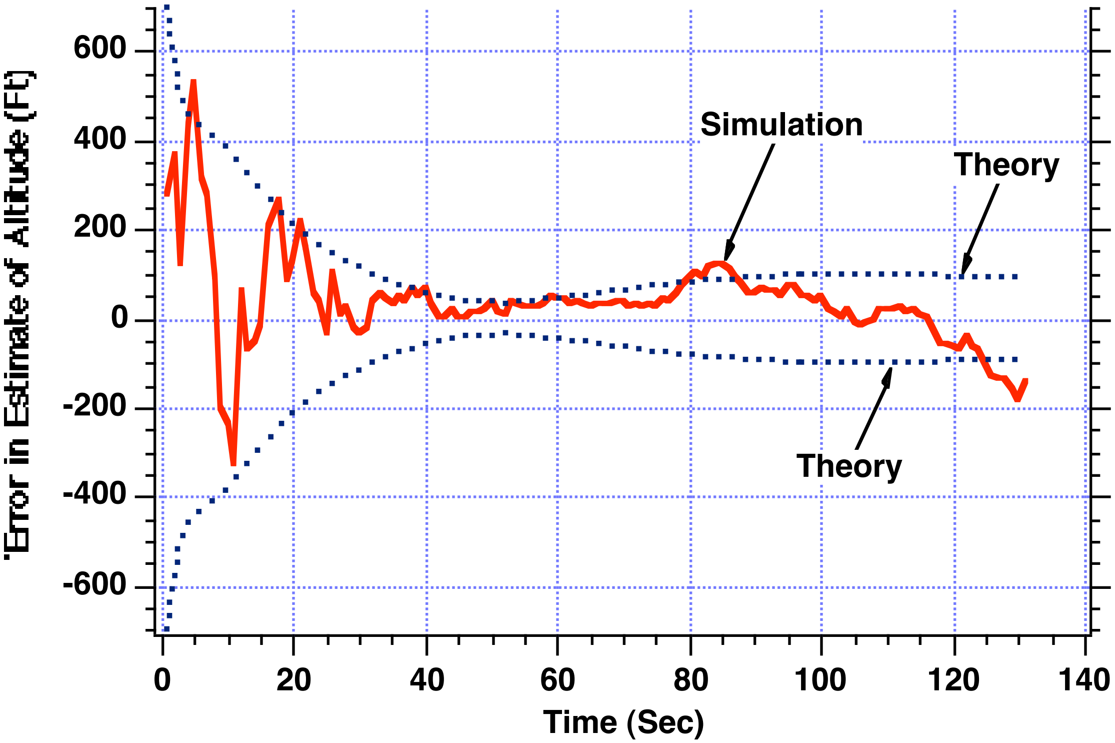
Extended Cartesian Kalman Filter's Projectile's Downrange Estimates Appear to Agree With Theory



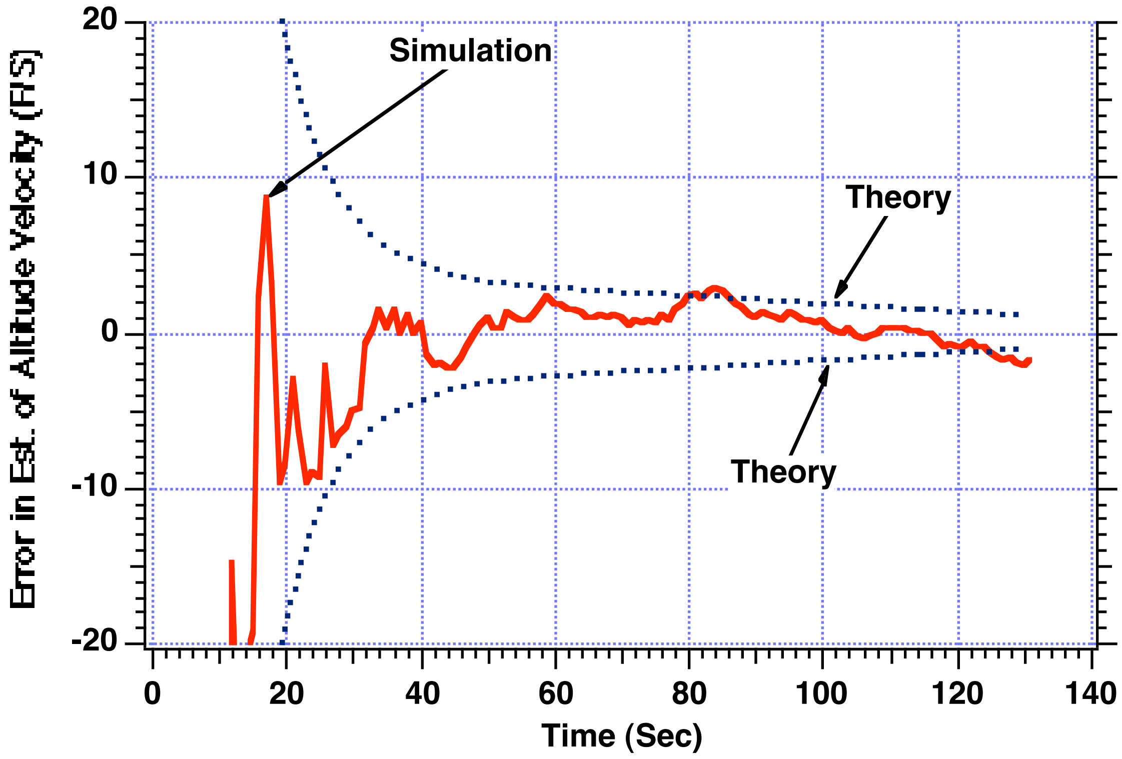
Extended Cartesian Kalman Filter's Projectile's Downrange Velocity Estimates Appear to Agree With Theory



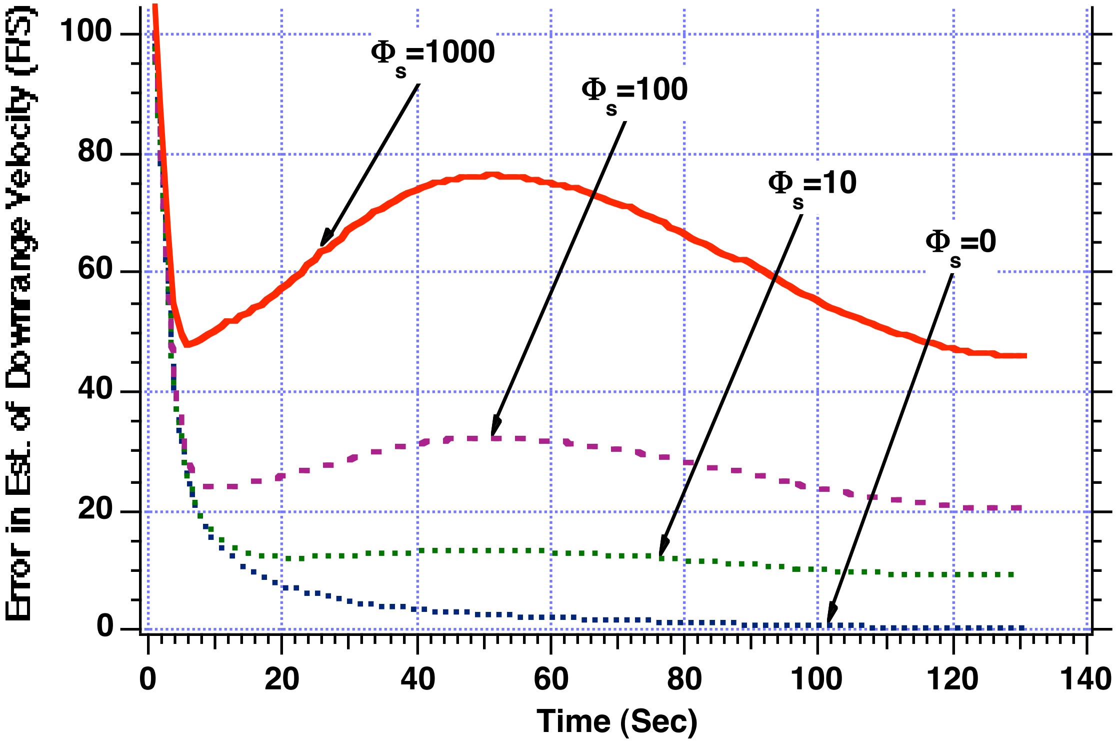
Extended Cartesian Kalman Filter's Projectile's Altitude Estimates Appear to Agree With Theory



Extended Cartesian Kalman Filter's Projectile's Altitude Velocity Estimates Appear to Agree With Theory



Errors in Estimate of Velocity Increase With Increasing Process Noise



$$Q_k = \begin{bmatrix} \frac{T_s^3\Phi_s}{3} & \frac{T_s^2\Phi_s}{2} & 0 & 0 \\ \frac{T_s^2\Phi_s}{2} & T_s\Phi_s & 0 & 0 \\ 0 & 0 & \frac{T_s^3\Phi_s}{3} & \frac{T_s^2\Phi_s}{2} \\ 0 & 0 & \frac{T_s^2\Phi_s}{2} & T_s\Phi_s \end{bmatrix}$$

Polar Coordinate System

Develop Polar Equations

Recall

$$\theta = \tan^{-1}\left[\frac{y_T - y_R}{x_T - x_R}\right]$$

$$r = \sqrt{(x_T - x_R)^2 + (y_T - y_R)^2}$$

Taking first derivative of angle

$$\dot{\theta} = \frac{1}{1 + \left(\frac{y_T - y_R}{x_T - x_R}\right)^2} \frac{(x_T - x_R)\,\dot{y}_T - (y_T - y_R)\,\dot{x}_T}{(x_T - x_R)^2}$$

Simplification yields

$$\dot{\theta} = \frac{(x_T - x_R)\,\dot{y}_T - (y_T - y_R)\,\dot{x}_T}{r^2}$$

Take first derivative of range

$$\dot{r} = \frac{1}{2}\left[\,(x_T - x_R)^2 \;+ (y_T - y_R)^2\right]^{-1/2}\left[2\,(x_T - x_R)\dot{x}_T + \;2\,(y_T - y_R)\dot{y}_T\right]$$

Which simplifies to

$$\dot{r} = \frac{(x_T - x_R)\dot{x}_T + \;(y_T - y_R)\dot{y}_T}{r}$$

Developing Angular Acceleration Formula

Taking second derivative of angle

$$\ddot{\theta} = \frac{r^2 \left[\dot{x}_T \dot{y}_T + (x_T - x_R)\, \ddot{y}_T - \dot{x}_T \dot{y}_T - (y_T - y_R)\, \ddot{x}_T\right] - \left[(x_T - x_R)\dot{y}_T - (y_T - y_R)\dot{y}_T\right] 2r\dot{r}}{r^4}$$

Which simplifies to

$$\ddot{\theta} = \frac{(x_T - x_R)\, \ddot{y}_T - (y_T - y_R)\, \ddot{x}_T - 2\dot{r}\dot{\theta}}{r^2}$$

Recognizing that

$$\ddot{x}_T = 0$$

$$\ddot{y}_T = -g$$

$$\cos\theta = \frac{x_T - x_R}{r}$$

$$\sin\theta = \frac{y_T - y_R}{r}$$

We get

$$\ddot{\theta} = \frac{-g \cos\theta - 2\dot{r}\dot{\theta}}{r}$$

Developing Range Acceleration Formula

Taking second derivative of range

$$\ddot{r} = \frac{r\left[\dot{x}_T\dot{x}_T + (x_T - x_R)\ddot{x}_T + \dot{y}_T\dot{y}_T + (y_T - y_R)\ddot{y}_T\right] - \left[(x_T - x_R)\dot{x}_T + (y_T - y_R)\dot{y}_T\right]\dot{r}}{r^2}$$

Which simplifies to

$$\ddot{r} = \frac{r^2\dot{\theta}^2 - gr\sin\theta}{r}$$

Differential Equation Summary

Polar differential equations

$$\ddot{\theta} = \frac{-g \cos \theta - 2\dot{r}\dot{\theta}}{r}$$

$$\ddot{r} = \frac{r^2\dot{\theta}^2 - gr \sin \theta}{r}$$

Polar initial conditions

$$\theta(0) = \tan^{-1}\left[\frac{y_T(0) - y_R}{x_T(0) - x_R}\right]$$

$$r(0) = \sqrt{(x_T(0) - x_R)^2 + (y_T(0) - y_R)^2}$$

$$\dot{\theta}(0) = \frac{(x_T(0) - x_R)\,\dot{y}_T(0) - (y_T(0) - y_R)\,\dot{x}_T(0)}{r(0)^2}$$

$$\dot{r}(0) = \frac{(x_T(0) - x_R)\dot{x}_T(0) + (y_T(0) - y_R)\dot{y}_T(0)}{r(0)}$$

Cartesian differential equations

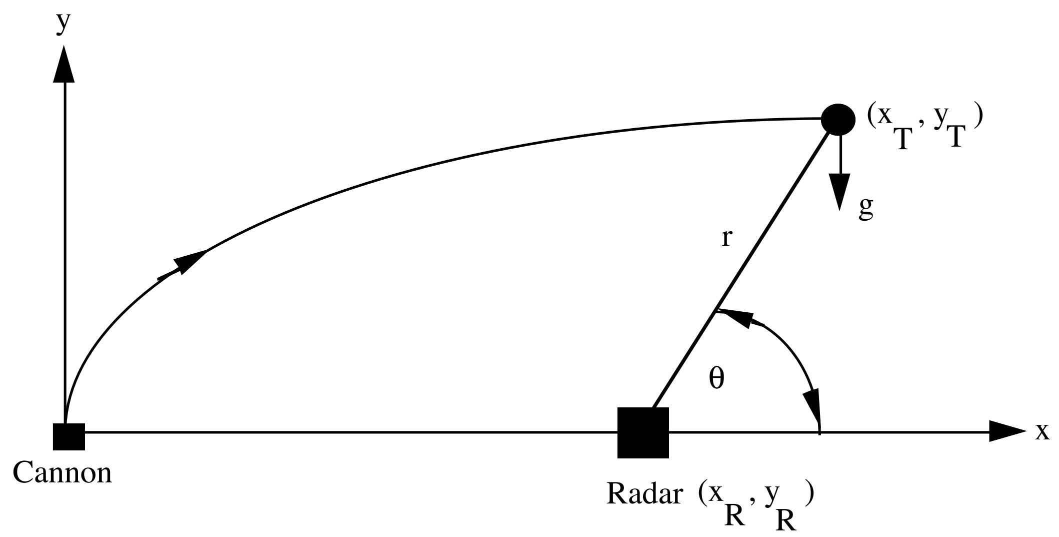
$$\ddot{x}_T = 0$$

$$\ddot{y}_T = -g$$

Cartesian initial conditions

$$x_T(0),\ y_T(0),\ \dot{x}_T(0),\ \dot{y}_T(0)$$

Going From Polar to Cartesian



Obtaining position

$$\hat{x}_T = r \cos\theta + x_R$$

$$\hat{y}_T = r \sin\theta + y_R$$

Obtaining velocity

$$\hat{\dot{x}}_T = \dot{r} \cos\theta - r\dot{\theta} \sin\theta$$

$$\hat{\dot{y}}_T = \dot{r} \sin\theta + r\dot{\theta} \cos\theta$$

FORTRAN Simulation That Compares Polar and Cartesian Differential Equations-1

```
IMPLICIT REAL*8(A-H,O-Z)
VT=3000.
GAMDEG=45.
G=32.2
TS=1.
XT=0.
YT=0.
XTD=VT*COS(GAMDEG/57.3)
YTD=VT*SIN(GAMDEG/57.3)
```

Cartesian initial conditions

```
XR=100000.
YR=0.
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
T=0.
S=0.
H=.001
THET=ATAN2((YT-YR),(XT-XR))
RT=SQRT((XT-XR)**2+(YT-YR)**2)
THETD=((XT-XR)*YTD-(YT-YR)*XTD)/RT**2
RTD=((XT-XR)*XTD+(YT-YR)*YTD)/RT
```

Polar initial conditions

```
WHILE(YT>=0.)
          XTOLD=XT
          XTDOLD=XTD
          YTOLD=YT
          YTDOLD=YTD
          THETOLD=THET
          THETDOLD=THETD
          RTOLD=RT
          RTDOLD=RTD
          XTDD=0.
          YTDD=-G
          THETDD=(-G*COS(THET)-2.*RTD*THETD)/RT
          RTDD=((RT*THETD)**2-G*RT*SIN(THET))/RT
```

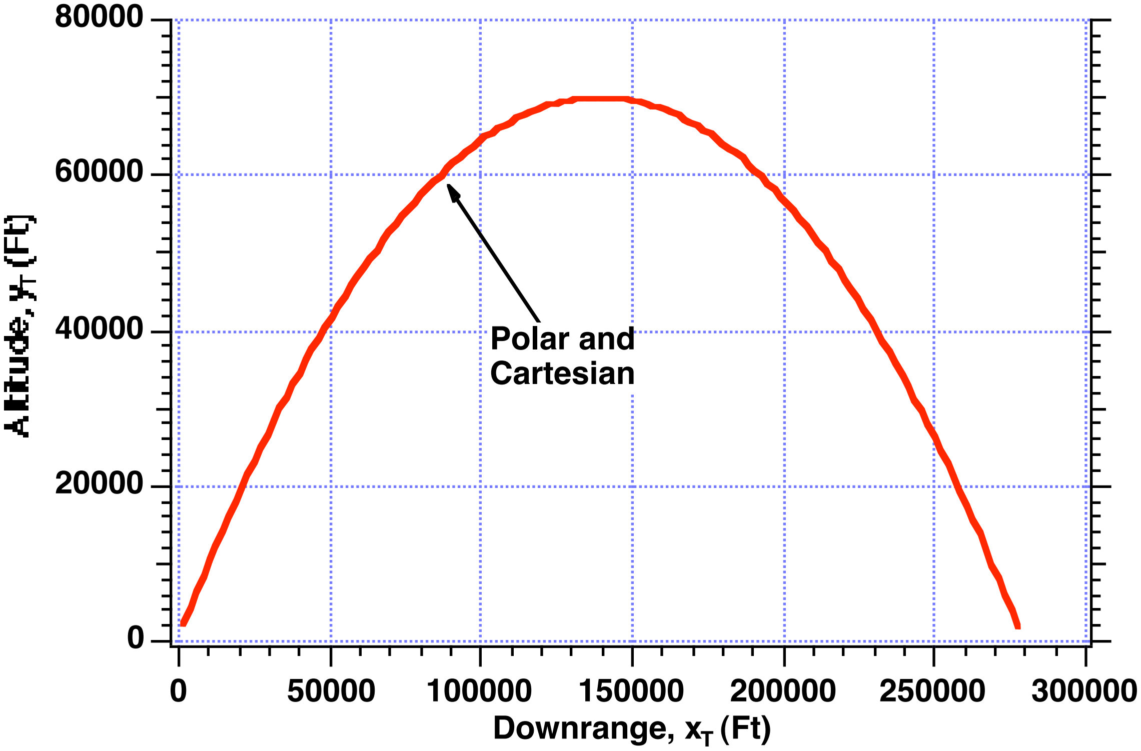
Second-order Runge-Kutta integration

FORTRAN Simulation That Compares Polar and Cartesian Differential Equations-2

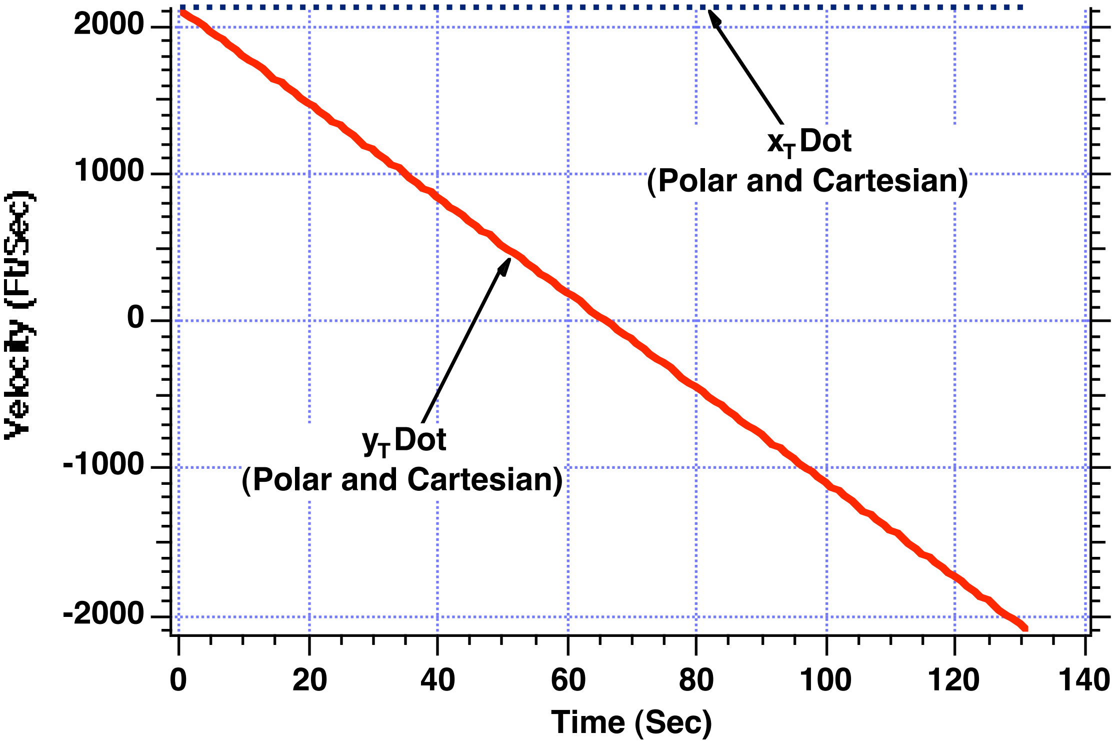
```
      XT=XT+H*XTD
      XTD=XTD+H*XTDD
      YT=YT+H*YTD
      YTD=YTD+H*YTDD
      THET=THET+H*THETD
      THETD=THETD+H*THETDD
      RT=RT+H*RTD
      RTD=RTD+H*RTDD
      T=T+H
      XTDD=0.
      YTDD=-G
      THETDD=(-G*COS(THET)-2.*RTD*THETD)/RT
      RTDD=((RT*THETD)**2-G*RT*SIN(THET))/RT
      XT=.5*(XTOLD+XT+H*XTD)
      XTD=.5*(XTDOLD+XTD+H*XTDD)
      YT=.5*(YTOLD+YT+H*YTD)
      YTD=.5*(YTDOLD+YTD+H*YTDD)
      THET=.5*(THETOLD+THET+H*THETD)
      THETD=.5*(THETDOLD+THETD+H*THETDD)
      RT=.5*(RTOLD+RT+H*RTD)
      RTD=.5*(RTDOLD+RTD+H*RTDD)
      S=S+H
      IF(S>=(TS-.00001))THEN
            S=0.
            XTH=RT*COS(THET)+XR
            YTH=RT*SIN(THET)+YR
            XTDH=RTD*COS(THET)-RT*SIN(THET)*THETD
            YTDH=RTD*SIN(THET)+RT*COS(THET)*THETD
            WRITE(9,*)T,XT,XTH,YT,YTH,XTD,XTDH,YTD,YTDH
            WRITE(1,*)T,XT,XTH,YT,YTH,XTD,XTDH,YTD,YTDH
      ENDIF
END DO
PAUSE
CLOSE(1)
END
```

Compare polar and Cartesian

Polar and Cartesian Differential Equations are Identical in Position



Polar and Cartesian Differential Equations are Identical in Velocity



Extended Polar Kalman Filter

Developing Polar Extended Kalman Filter Equations

Our model of the real world is now nonlinear

$$\ddot{\theta} = \frac{-g \cos \theta - 2\dot{r}\dot{\theta}}{r}$$

$$\ddot{r} = \frac{r^2\dot{\theta}^2 - gr \sin \theta}{r}$$

Linearized model of real world assuming zero process noise

$$\begin{bmatrix} \Delta\dot{\theta} \\ \Delta\ddot{\theta} \\ \Delta\dot{r} \\ \Delta\ddot{r} \end{bmatrix} = \begin{bmatrix} \frac{\partial\dot{\theta}}{\partial\theta} & \frac{\partial\dot{\theta}}{\partial\dot{\theta}} & \frac{\partial\dot{\theta}}{\partial r} & \frac{\partial\dot{\theta}}{\partial\dot{r}} \\ \frac{\partial\ddot{\theta}}{\partial\theta} & \frac{\partial\ddot{\theta}}{\partial\dot{\theta}} & \frac{\partial\ddot{\theta}}{\partial r} & \frac{\partial\ddot{\theta}}{\partial\dot{r}} \\ \frac{\partial\dot{r}}{\partial\theta} & \frac{\partial\dot{r}}{\partial\dot{\theta}} & \frac{\partial\dot{r}}{\partial r} & \frac{\partial\dot{r}}{\partial\dot{r}} \\ \frac{\partial\ddot{r}}{\partial\theta} & \frac{\partial\ddot{r}}{\partial\dot{\theta}} & \frac{\partial\ddot{r}}{\partial r} & \frac{\partial\ddot{r}}{\partial\dot{r}} \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\dot{\theta} \\ \Delta r \\ \Delta\dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{\partial\ddot{\theta}}{\partial\theta} & \frac{\partial\ddot{\theta}}{\partial\dot{\theta}} & \frac{\partial\ddot{\theta}}{\partial r} & \frac{\partial\ddot{\theta}}{\partial\dot{r}} \\ 0 & 0 & 0 & 1 \\ \frac{\partial\ddot{r}}{\partial\theta} & \frac{\partial\ddot{r}}{\partial\dot{\theta}} & \frac{\partial\ddot{r}}{\partial r} & \frac{\partial\ddot{r}}{\partial\dot{r}} \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\dot{\theta} \\ \Delta r \\ \Delta\dot{r} \end{bmatrix}$$

Systems dynamics matrix

Developing Fundamental Matrix-1

Evaluating partial derivatives yields

$$\mathbf{F} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{g\sin\theta}{r} & \frac{-2\dot{r}}{r} & \frac{g\cos\theta + 2\dot{\theta}\dot{r}}{r^2} & \frac{-2\dot{\theta}}{r} \\ 0 & 0 & 0 & 1 \\ -g\cos\theta & 2r\dot{\theta} & \dot{\theta}^2 & 0 \end{bmatrix}$$

Two term Taylor series expansion

$$\mathbf{\Phi}(t) = \mathbf{I} + \mathbf{F}\,t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & t & 0 & 0 \\ \frac{g\sin\theta}{r}t & \frac{-2\dot{r}}{r}t & \frac{g\cos\theta + 2\dot{\theta}\dot{r}}{r^2}t & \frac{-2\dot{\theta}}{r}t \\ 0 & 0 & 0 & t \\ -g\cos\theta t & 2r\dot{\theta}t & \dot{\theta}^2 t & 0 \end{bmatrix}$$

Developing Fundamental Matrix-2

Simplification yields

$$\boldsymbol{\Phi}(t) = \begin{bmatrix} 1 & t & 0 & 0 \\ \frac{g\sin\theta}{r}t & 1 - \frac{2\dot{r}}{r}t & \frac{g\cos\theta + 2\dot{\theta}\dot{r}}{r^2}t & \frac{-2\dot{\theta}}{r}t \\ 0 & 0 & 1 & t \\ -g\cos\theta t & 2r\dot{\theta}t & \dot{\theta}^2 t & 1 \end{bmatrix}$$

Or in discrete form

$$\boldsymbol{\Phi}_{\mathbf{k}} = \begin{bmatrix} 1 & T_s & 0 & 0 \\ \frac{g\sin\theta}{r}T_s & 1 - \frac{2\dot{r}}{r}T_s & \frac{g\cos\theta + 2\dot{\theta}\dot{r}}{r^2}T_s & \frac{-2\dot{\theta}}{r}T_s \\ 0 & 0 & 1 & T_s \\ -g\cos\theta T_s & 2r\dot{\theta}T_s & \dot{\theta}^2 T_s & 1 \end{bmatrix}$$

Measurement Equation is Linear in Polar System

Measurement equation

$$\begin{bmatrix} \theta^* \\ r^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \dot{\theta} \\ r \\ \dot{r} \end{bmatrix} + \begin{bmatrix} v_\theta \\ v_r \end{bmatrix}$$

Measurement matrix

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

Discrete measurement noise matrix

$$\mathbf{R_k} = E(\mathbf{v_k} \mathbf{v_k^T}) \longrightarrow \mathbf{R_k} = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix}$$

We now are able to solve Riccati equations

Additional Equations For Comparing Polar and Cartesian Extended Kalman Filters-1

Recall we can derive Cartesian quantities from polar quantities

$$x_T = r\cos\theta + x_R$$

$$y_T = r\sin\theta + y_R$$

$$\dot{x}_T = \dot{r}\cos\theta - r\dot{\theta}\sin\theta$$

$$\dot{y}_T = \dot{r}\sin\theta + r\dot{\theta}\cos\theta$$

Using chain rule from calculus

$$\Delta x_T = \frac{\partial x_T}{\partial \theta}\Delta\theta + \frac{\partial x_T}{\partial \dot{\theta}}\Delta\dot{\theta} + \frac{\partial x_T}{\partial r}\Delta r + \frac{\partial x_T}{\partial \dot{r}}\Delta\dot{r}$$

$$\Delta \dot{x}_T = \frac{\partial \dot{x}_T}{\partial \theta}\Delta\theta + \frac{\partial \dot{x}_T}{\partial \dot{\theta}}\Delta\dot{\theta} + \frac{\partial \dot{x}_T}{\partial r}\Delta r + \frac{\partial \dot{x}_T}{\partial \dot{r}}\Delta\dot{r}$$

$$\Delta y_T = \frac{\partial y_T}{\partial \theta}\Delta\theta + \frac{\partial y_T}{\partial \dot{\theta}}\Delta\dot{\theta} + \frac{\partial y_T}{\partial r}\Delta r + \frac{\partial y_T}{\partial \dot{r}}\Delta\dot{r}$$

$$\Delta \dot{y}_T = \frac{\partial \dot{y}_T}{\partial \theta}\Delta\theta + \frac{\partial \dot{y}_T}{\partial \dot{\theta}}\Delta\dot{\theta} + \frac{\partial \dot{y}_T}{\partial r}\Delta r + \frac{\partial \dot{y}_T}{\partial \dot{r}}\Delta\dot{r}$$

Additional Equations For Comparing Polar and Cartesian Extended Kalman Filters-2

Evaluating partial derivatives yields

$$\Delta x_T = -r\sin\theta\ \Delta\theta + \cos\theta\ \Delta r$$

$$\Delta \dot{x}_T = (-\dot{r}\sin\theta - r\dot{\theta}\cos\theta)\ \Delta\theta - r\sin\theta\ \Delta\dot{\theta} - \dot{\theta}\sin\theta\ \Delta r + \cos\theta\ \Delta\dot{r}$$

$$\Delta y_T = r\cos\theta\ \Delta\theta + \sin\theta\ \Delta r$$

$$\Delta \dot{y}_T = (\dot{r}\cos\theta - r\dot{\theta}\sin\theta)\ \Delta\theta + r\cos\theta\ \Delta\dot{\theta} + \dot{\theta}\cos\theta\ \Delta r + \sin\theta\ \Delta\dot{r}$$

Or in state space form

$$\begin{bmatrix} \Delta x_T \\ \Delta \dot{x}_T \\ \Delta y_T \\ \Delta \dot{y}_T \end{bmatrix} = \underbrace{\begin{bmatrix} -r\sin\theta & 0 & \cos\theta & 0 \\ -\dot{r}\sin\theta - r\dot{\theta}\cos\theta & -r\sin\theta & -\dot{\theta}\sin\theta & \cos\theta \\ r\cos\theta & 0 & \sin\theta & 0 \\ \dot{r}\cos\theta & r\cos\theta & \dot{\theta}\cos\theta & \sin\theta \end{bmatrix}}_{\textbf{A}} \begin{bmatrix} \Delta\theta \\ \Delta\dot{\theta} \\ \Delta r \\ \Delta\dot{r} \end{bmatrix}$$

Relationship between covariance matrices

$$\mathbf{P}_{\mathbf{CART}} = \mathbf{A}\mathbf{P}_{\mathbf{POL}}\mathbf{A}^{\mathbf{T}}$$

Extended Polar Kalman Filter

Filtering equations

$$\hat{\theta}_k = \bar{\theta}_k + K_{11_k}(\theta^*_k - \bar{\theta}_k) + K_{12_k}(r^*_k - \bar{r}_k)$$

$$\hat{\dot{\theta}}_k = \bar{\dot{\theta}}_k + K_{21_k}(\theta^*_k - \bar{\theta}_k) + K_{22_k}(r^*_k - \bar{r}_k)$$

$$\hat{r}_k = \bar{r}_k + K_{31_k}(\theta^*_k - \bar{\theta}_k) + K_{32_k}(r^*_k - \bar{r}_k)$$

$$\hat{\dot{r}}_k = \bar{\dot{r}}_k + K_{41_k}(\theta^*_k - \bar{\theta}_k) + K_{42_k}(r^*_k - \bar{r}_k)$$

Where barred quantities are obtained by numerical integration (more computationally expensive)

True BASIC Polar Extended Kalman Filter-1

```
OPTION NOLET
REM UNSAVE "DATFIL"
REM UNSAVE "COVFIL"
REM UNSAVE "COVFIL2"
OPEN #1:NAME "DATFIL",ACCESS OUTPUT,CREATE NEW, ORGANIZATION TEXT
OPEN #2:NAME "COVFIL",ACCESS OUTPUT,CREATE NEW, ORGANIZATION TEXT
OPEN #3:NAME "COVFIL2",ACCESS OUTPUT,CREATE NEW, ORGANIZATION TEXT
SET #1: MARGIN 1000
SET #2: MARGIN 1000
SET #3: MARGIN 1000
DIM P(4,4),M(4,4),GAIN(4,2),PHI(4,4),PHIT(4,4),PHIP(4,4)
DIM Q(4,4),HMAT(2,4),HM(2,4),MHT(4,2),PHIPPHIT(4,4),AT(4,4)
DIM RMAT(2,2),HMHTR(2,2),HMHTRINV(2,2),A(4,4),PNEW(4,4)
DIM HMHT(2,2),HT(4,2),KH(4,4),IDNP(4,4),IKH(4,4),FTS(4,4)
DIM AP(4,4)
SIGTH=.01
SIGR=100.
TS=1.
G=32.2
VT=3000.
GAMDEG=45.
```

```
XT=0.
YT=0.
XTD=VT*COS(GAMDEG/57.3)
YTD=VT*SIN(GAMDEG/57.3)
```

Initial conditions for Cartesian equations

```
XR=100000.
YR=0.
```

```
XTH=XT+1000.
YTH=YT-1000.
XTDH=XTD-100.
YTDH=YTD+100.
```

Original initial state estimates for Cartesian

```
CALL ATAN2((YT-YR), (XT-XR)+.001, TH)
R=SQR((XT-XR)^2+(YT-YR)^2)
THD=((XT-XR)*YTD-(YT-YR)*XTD)/R^2
RD=((XT-XR)*XTD+(YT-YR)*YTD)/R
```

Initial conditions for polar equations

```
CALL ATAN2((YTH-YR), (XTH-XR)+.001, THH)
RH=SQR((XTH-XR)^2+(YTH-YR)^2)
THDH=((XTH-XR)*YTDH-(YTH-YR)*XTDH)/RH^2
RDH=((XTH-XR)*XTDH+(YTH-YR)*YTDH)/RH
```

Initial state estimates for polar filter

```
ORDER=4
TF=100.
T=0.
S=0.
H=.001
HP=.001
```

True BASIC Polar Extended Kalman Filter-2

```
MAT PHI=ZER(ORDER,ORDER)
MAT P=ZER(ORDER,ORDER)
MAT IDNP=IDN(ORDER,ORDER)
MAT Q=ZER(ORDER,ORDER)
MAT FTS=ZER(ORDER,ORDER)
MAT A=ZER(ORDER,ORDER)
P(1,1)=(TH-THH)^2
P(2,2)=(THD-THDH)^2
P(3,3)=(R-RH)^2
P(4,4)=(RD-RDH)^2
```

Initial covariance matrix

```
RMAT(1,1)=SIGTH^2
RMAT(1,2)=0.
RMAT(2,1)=0.
RMAT(2,2)=SIGR^2
```

Measurement noise matrix

```
HMAT(1,1)=1.
HMAT(1,2)=0.
HMAT(1,3)=0.
HMAT(1,4)=0.
HMAT(2,1)=0.
HMAT(2,2)=0.
HMAT(2,3)=1.
HMAT(2,4)=0.
```

Measurement matrix

```
DO WHILE YT>=0.
        THOLD=TH
        THDOLD=THD
        ROLD=R
        RDOLD=RD
        THDD=(-G*R*COS(TH)-2.*THD*R*RD)/R^2
        RDD=(R*R*THD*THD-G*R*SIN(TH))/R
        TH=TH+H*THD
        THD=THD+H*THDD
        R=R+H*RD
        RD=RD+H*RDD
        T=T+H
        THDD=(-G*R*COS(TH)-2.*THD*R*RD)/R^2
        RDD=(R*R*THD*THD-G*R*SIN(TH))/R
        TH=.5*(THOLD+TH+H*THD)
        THD=.5*(THDOLD+THD+H*THDD)
        R=.5*(ROLD+R+H*RD)
        RD=.5*(RDOLD+RD+H*RDD)
        S=S+H
```

Second-order Runge-Kutta integration of polar differential equations

True BASIC Polar Extended Kalman Filter-3

```
IF S>=(TS-.00001) THEN
        S=0.
        FTS(1,2)=1.*TS
        FTS(2,1)=G*SIN(THH)*TS/RH
        FTS(2,2)=-2.*RDH*TS/RH
        FTS(2,3)=(G*COS(THH)+2.*THDH*RDH)*TS/RH^2
        FTS(2,4)=-2.*THDH*TS/RH
        FTS(3,4)=1.*TS
        FTS(4,1)=-G*COS(THH)*TS
        FTS(4,2)=2.*RH*THDH*TS
        FTS(4,3)=(THDH^2)*TS
        MAT PHI=FTS+IDNP
        MAT HT=TRN(HMAT)
        MAT PHIT=TRN(PHI)
        MAT PHIP=PHI*P
        MAT PHIPPHIT=PHIP*PHIT
        MAT M=PHIPPHIT+Q
        MAT HM=HMAT*M
        MAT HMHT=HM*HT
        MAT HMHTR=HMHT+RMAT
        MAT HMHTRINV=INV(HMHTR)
        MAT MHT=M*HT
        MAT GAIN=MHT*HMHTRINV
        MAT KH=GAIN*HMAT
        MAT IKH=IDNP-KH
        MAT P=IKH*M
        CALL GAUSS(THNOISE,SIGTH)
        CALL GAUSS(RNOISE,SIGR)
        CALL PROJECT(T,TS,THH,THDH,RH,RDH,THB,THDB,RB,RDB,HP)
        RES1=TH+THNOISE-THB
        RES2=R+RNOISE-RB
        THH=THB+GAIN(1,1)*RES1+GAIN(1,2)*RES2
        THDH=THDB+GAIN(2,1)*RES1+GAIN(2,2)*RES2
        RH=RB+GAIN(3,1)*RES1+GAIN(3,2)*RES2
        RDH=RDB+GAIN(4,1)*RES1+GAIN(4,2)*RES2
```

Fundamental matrix

Riccati equations

Propagate estimates

Filter

True BASIC Polar Extended Kalman Filter-4

```
ERRTH=TH-THH
SP11=SQR(P(1,1))
ERRTHD=THD-THDH
SP22=SQR(P(2,2))
ERRR=R-RH
SP33=SQR(P(3,3))
ERRRD=RD-RDH
SP44=SQR(P(4,4))
```

Polar errors in estimates

```
XT=R*COS(TH)+XR
YT=R*SIN(TH)+YR
XTD=RD*COS(TH)-R*THD*SIN(TH)
YTD=RD*SIN(TH)+R*THD*COS(TH)
```

Actual Cartesian states

```
XTH=RH*COS(THH)+XR
YTH=RH*SIN(THH)+YR
XTDH=RDH*COS(THH)-RH*THDH*SIN(THH)
YTDH=RDH*SIN(THH)+RH*THDH*COS(THH)
```

Estimated Cartesian states

```
A(1,1)=-RH*SIN(THH)
A(1,3)=COS(THH)
A(2,1)=-RDH*SIN(THH)-RH*THDH*COS(THH)
A(2,2)=-RH*SIN(THH)
A(2,3)=-THDH*SIN(THH)
A(2,4)=COS(THH)
A(3,1)=RH*COS(THH)
A(3,3)=SIN(THH)
A(4,1)=RDH*COS(THH)-RH*SIN(THH)*THDH
A(4,2)=RH*COS(THH)
A(4,3)=THDH*COS(THH)
A(4,4)=SIN(THH)
```

Transformation matrix

```
MAT AT=TRN(A)
MAT AP=A*P
MAT PNEW=AP*AT
```

Cartesian covariance matrix

```
ERRXT=XT-XTH
SP11P=SQR(PNEW(1,1))
ERRXTD=XTD-XTDH
SP22P=SQR(PNEW(2,2))
ERRYT=YT-YTH
SP33P=SQR(PNEW(3,3))
ERRYTD=YTD-YTDH
SP44P=SQR(PNEW(4,4))
```

Cartesian errors in estimates

```
                                        PRINT T,R,RH,RD,RDH,TH,THH,THD,THDH
                                        PRINT #1:T,R,RH,RD,RDH,TH,THH,THD,THDH
                                        PRINT #2:T,ERRTH,SP11,-SP11,ERRTHD,SP22,-SP22,ERRR,SP33,-SP33,ERRRD,SP44,-SP44
                                        PRINT #3:T,ERRXT,SP11P,-SP11P,ERRXTD,SP22P,-SP22P,ERRYT,SP33P,-SP33P,ERRYTD,SP44P,-
SP44P
            END IF
LOOP
CLOSE #1
CLOSE #2
CLOSE #3
END
```

True BASIC Polar Extended Kalman Filter-5

```
SUB ATAN2 (Y, X, Z)
IF X < 0 THEN
        IF Y < 0 THEN
                LET Z = ATN(Y / X) - 3.14159
        ELSE
                LET Z = ATN(Y / X) + 3.14159
        END IF
ELSE
        LET Z = ATN(Y / X)
END IF
END SUB
```

Arc tangent which is good in all quadrants

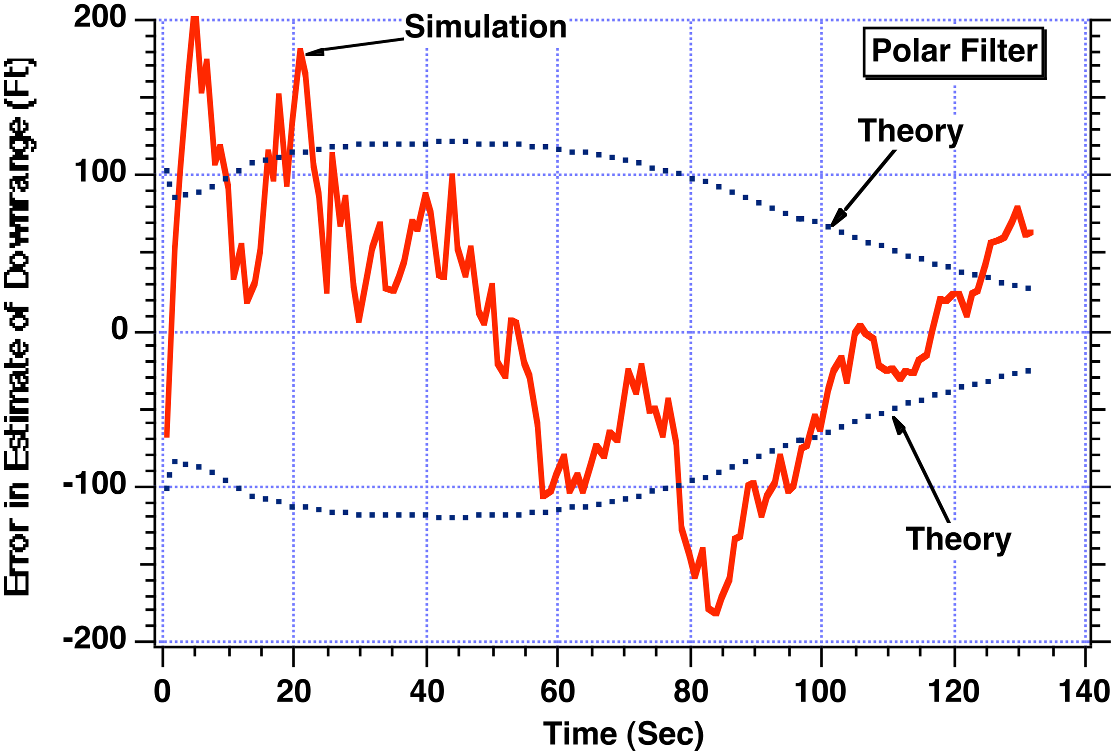
```
SUB PROJECT(TP,TS,THP,THDP,RP,RDP,THH,THDH,RH,RDH,HP)
T=0.
G=32.2
TH=THP
THD=THDP
R=RP
RD=RDP
H=HP
DO WHILE T<=(TS-.0001)
        THDD=(-G*R*COS(TH)-2.*THD*R*RD)/R^2
        RDD=(R*R*THD*THD-G*R*SIN(TH))/R
        THD=THD+H*THDD
        TH=TH+H*THD
        RD=RD+H*RDD
        R=R+H*RD
        T=T+H
LOOP
RH=R
RDH=RD
THH=TH
THDH=THD
END SUB
```

Propagate polar states ahead one sampling interval using Euler integration

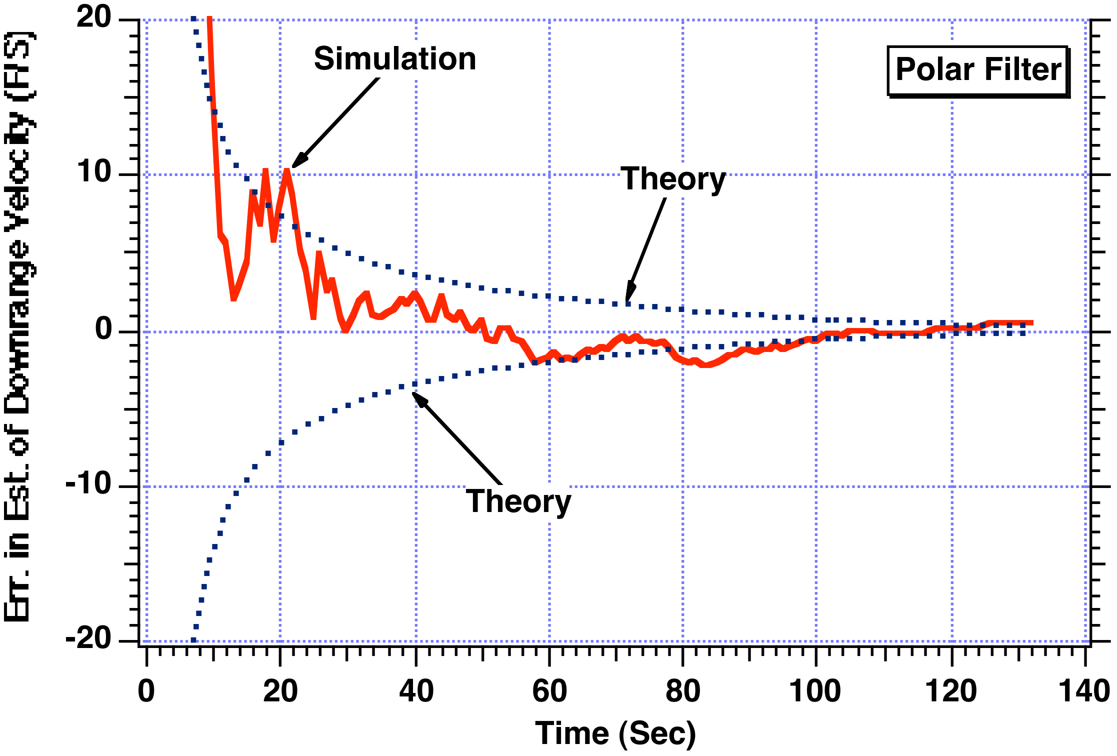
```
SUB GAUSS(X,SIG)
LET X=RND+RND+RND+RND+RND+RND-3
LET X=1.414*X*SIG
END SUB
```

Gaussian noise routine

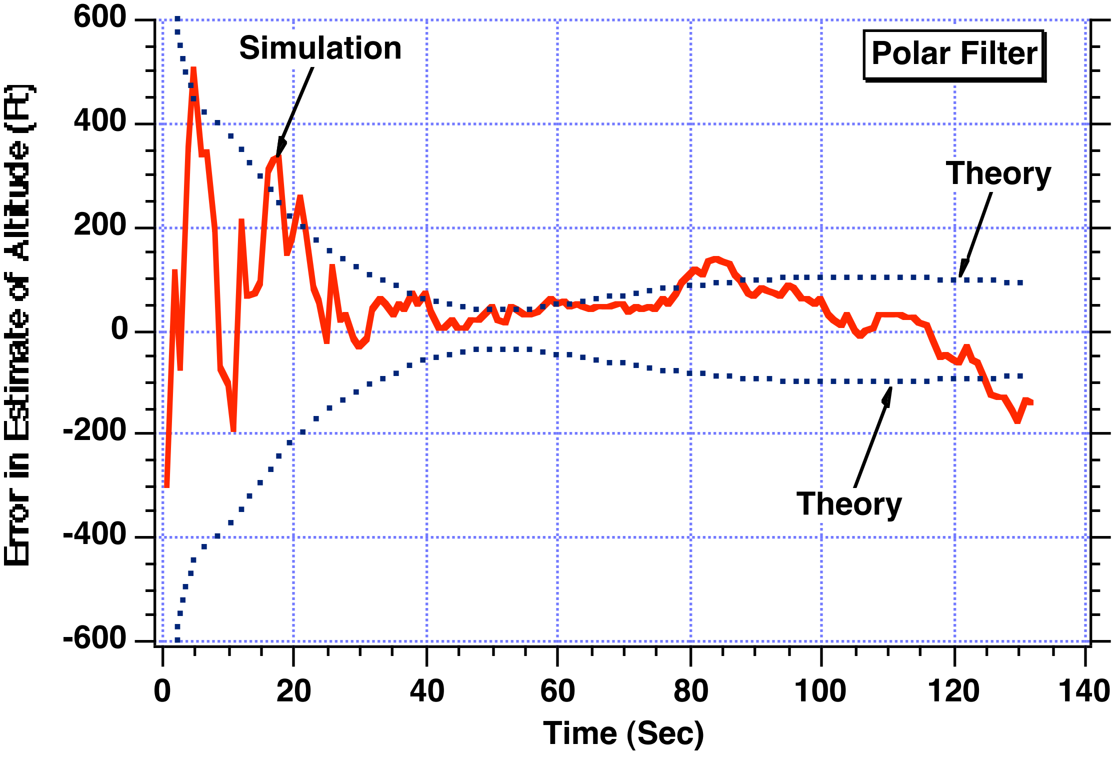
Polar and Cartesian Extended Kalman Filters Yield Similar Results for Downrange Estimates



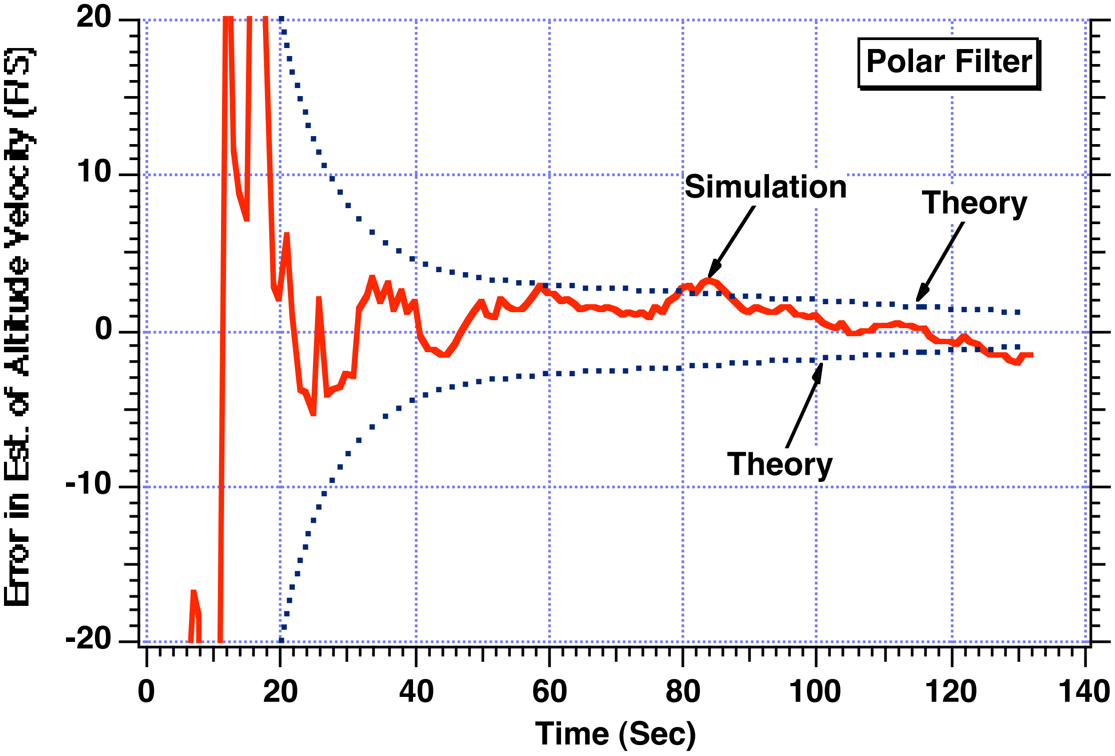
Polar and Cartesian Extended Kalman Filters Yield Similar Results for Downrange Velocity Estimates



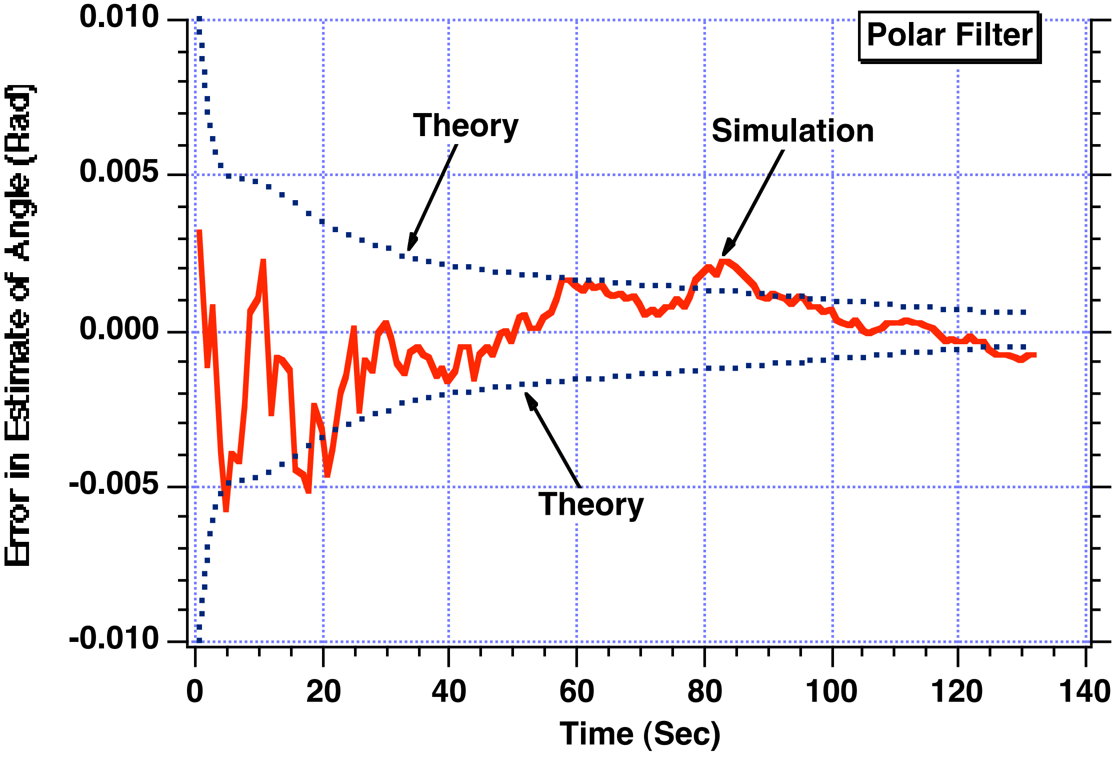
Polar and Cartesian Extended Kalman Filters Yield Similar Results for Altitude Estimates



Polar and Cartesian Extended Kalman Filters Yield Similar Results for Altitude Velocity Estimates



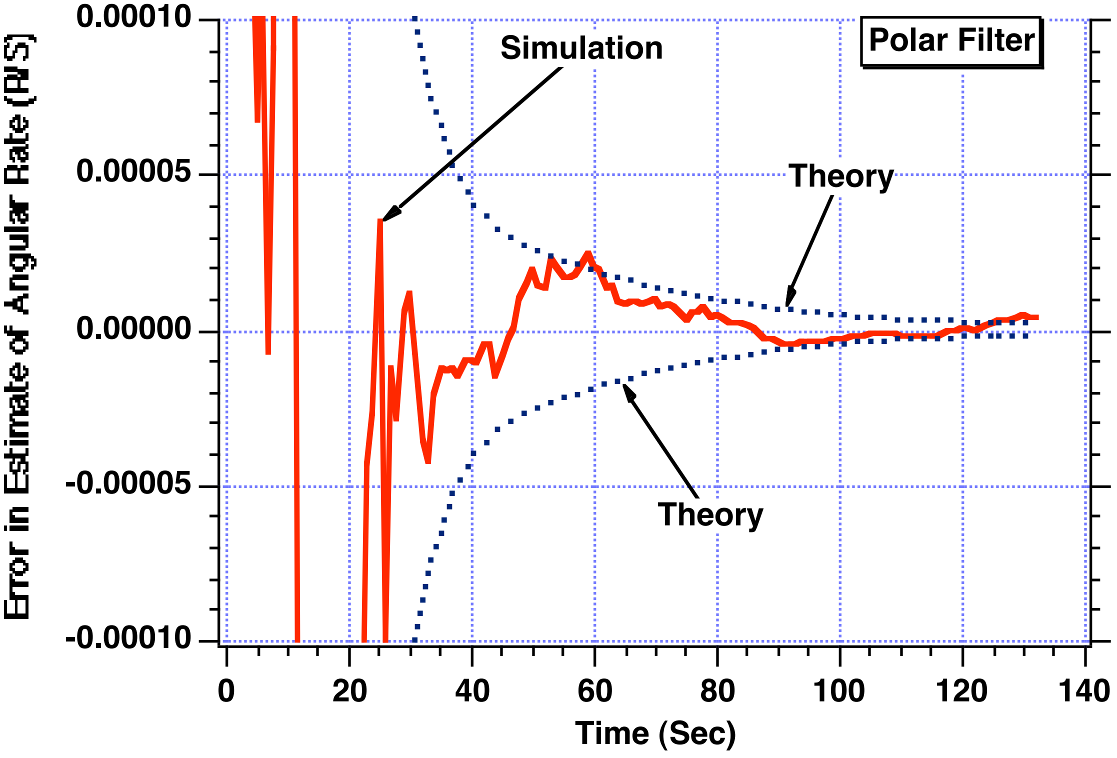
Error in the Estimate of Angle Indicates That Extended Polar Kalman Filter Appears to be Working Properly



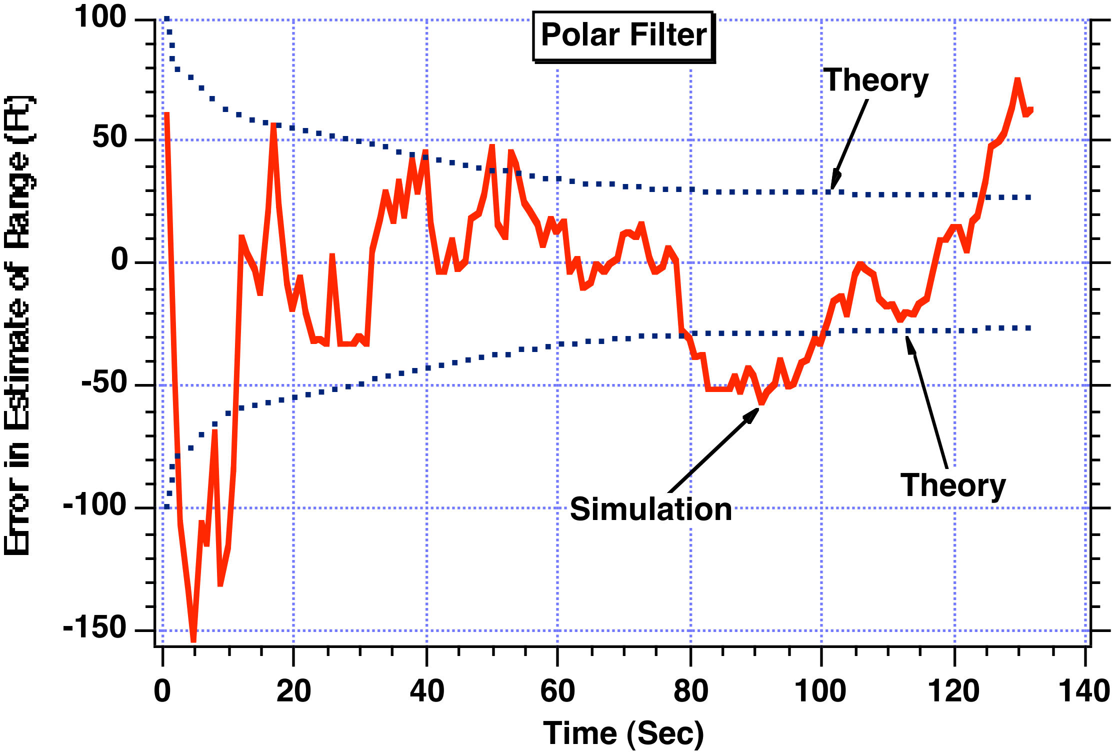
States

$$\begin{bmatrix} \theta \\ \dot{\theta} \\ r \\ \dot{r} \end{bmatrix}$$

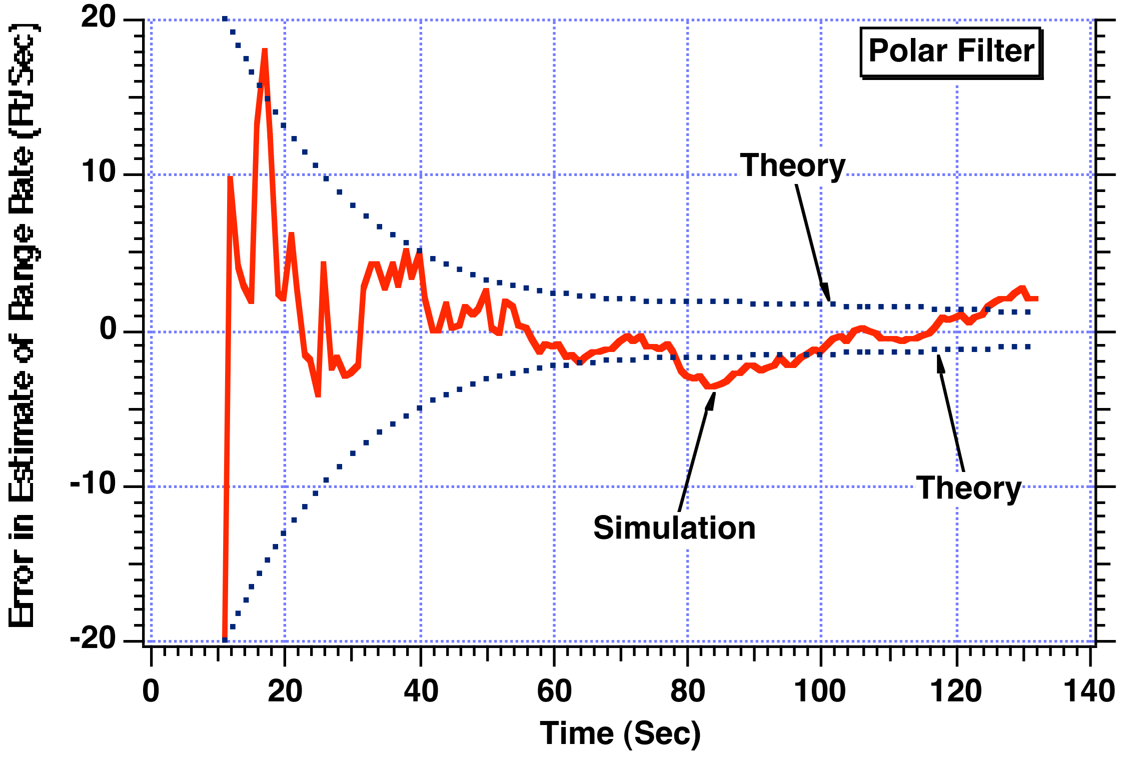
Error in the Estimate of Angle Rate Indicates That Extended Polar Kalman Filter Appears to be Working Properly



Error in the Estimate of Range Indicates That Extended Polar Kalman Filter Appears to be Working Properly



Error in the Estimate of Range Rate Indicates That Extended Polar Kalman Filter Appears to be Working Properly



Using Linear Decoupled Polynomial Kalman Filters

Making Measurement Noise Appear to be Linearly Related to States-1

In Cartesian frame model of real world is linear but measurements are nonlinear

Recall

$$x_T = r\cos\theta\ + x_R$$

$$y_T = r\sin\theta\ + y_R$$

Find total differential from calculus

$$\Delta x_T = \frac{\partial x_T}{\partial r}\Delta r + \frac{\partial x_T}{\partial \theta}\Delta\theta = \cos\theta\Delta r - r\sin\theta\Delta\theta$$

$$\Delta y_T = \frac{\partial y_T}{\partial r}\Delta r + \frac{\partial y_T}{\partial \theta}\Delta\theta = \sin\theta\Delta r + r\cos\theta\Delta\theta$$

Square both equations

$$\Delta x_T^2 = \cos^2\theta\Delta r^2 - 2r\sin\theta\cos\theta\Delta r\Delta\theta\ + r^2\sin^2\theta\Delta\theta^2$$

$$\Delta y_T^2 = \sin^2\theta\Delta r^2 + 2r\sin\theta\cos\theta\Delta r\Delta\theta\ + r^2\cos^2\theta\Delta\theta^2$$

Making Measurement Noise Appear to be Linearly Related to States-2

Taking expectations of both sides assuming range and angle are not correlated

$$E(\Delta x_T^2) = \cos^2\theta E(\Delta r^2) + r^2\sin^2\theta E(\Delta\theta^2)$$

$$E(\Delta y_T^2) = \sin^2\theta E(\Delta r^2) + r^2\cos^2\theta E(\Delta\theta^2)$$

Since

$$\sigma_{x_T}^2 = E(\Delta x_T^2)$$

$$\sigma_{y_T}^2 = E(\Delta y_T^2)$$

$$\sigma_r^2 = E(\Delta r^2)$$

$$\sigma_\theta^2 = E(\Delta\theta^2)$$

We can say

$$\sigma_{x_T}^2 = \cos^2\theta\sigma_r^2 + r^2\sin^2\theta\sigma_\theta^2$$

$$\sigma_{y_T}^2 = \sin^2\theta\sigma_r^2 + r^2\cos^2\theta\sigma_\theta^2$$

*We are pretending noise is on x and y rather than r and θ

Important Matrices in Downrange Channel-1

Model of real world

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

Process noise matrix

$$\mathbf{Q} = \begin{bmatrix} 0 & 0 \\ 0 & \Phi_s \end{bmatrix}$$

Systems dynamics matrix

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

For this F we have already seen that

$$\mathbf{\Phi(t)} = \begin{bmatrix} 0 & t \\ 0 & 1 \end{bmatrix}$$

Discrete fundamental matrix

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

Important Matrices in Downrange Channel-2

Discrete process noise matrix can be found from

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

We have already seen that

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

Due to our use of pseudonoise measurement equation is linear

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

Measurement matrix

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

Measurement noise matrix is a scalar

$$\mathbf{R_k} = \sigma_{x_T}^2 \longrightarrow \sigma_{x_T}^2 = \cos^2\theta\sigma_r^2 + r^2\sin^2\theta\sigma_\theta^2$$

From radar

Downrange Linear Polynomial Kalman Filter

Recall

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

Substituting matrices yields

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

Multiplying terms out yields

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

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

Important Matrices in Altitude Channel

Model of real world

$$\dot{\mathbf{x}} = \mathbf{Fx} + \mathbf{Gu} + \mathbf{w}$$

$$\begin{bmatrix}\dot{y}_T \\ \ddot{y}_T\end{bmatrix} = \begin{bmatrix}0 & 1 \\ 0 & 0\end{bmatrix}\begin{bmatrix}y_T \\ \dot{y}_T\end{bmatrix} + \begin{bmatrix}0 \\ -g\end{bmatrix} + \begin{bmatrix}0 \\ u_s\end{bmatrix}$$

Disturbance or control vector

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

Finding discrete disturbance vector

$$\mathbf{G_k} = \int_0^{T_s} \mathbf{\Phi G}\, d\tau = \int_0^{T_s} \begin{bmatrix}0 & \tau \\ 0 & 1\end{bmatrix}\begin{bmatrix}0 \\ -g\end{bmatrix} d\tau = \begin{bmatrix}-.5T_s^2 g \\ -T_s g\end{bmatrix}$$

Measurement equation

$$y_T^* = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}y_T \\ \dot{y}_T\end{bmatrix} + v_y$$

Measurement noise matrix is scalar

$$\mathbf{R_k} = \sigma_{y_T}^2 \longrightarrow \sigma_{y_T}^2 = \sin^2\theta\sigma_r^2 + r^2\cos^2\theta\sigma_\theta^2$$

***Matrices for Riccati equations are identical in both channels except for measurement noise**

Altitude Linear Polynomial Kalman Filter

Recall

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

Substituting matrices yields

$$\begin{bmatrix} y_{T_k} \\ \hat{\dot{y}}_{T_k} \end{bmatrix} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_{T_{k-1}} \\ \hat{\dot{y}}_{T_{k-1}} \end{bmatrix} - \begin{bmatrix} .5gT_s^2 \\ gT_s \end{bmatrix} + \begin{bmatrix} K_{1_k} \\ K_{2_k} \end{bmatrix} \left[y^*_{T_k} - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_{T_{k-1}} \\ \hat{\dot{y}}_{T_{k-1}} \end{bmatrix} + \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} .5gT_s^2 \\ gT_s \end{bmatrix} \right]$$

Multiplying out terms

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

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

MATLAB Version of Two Decoupled Polynomial Linear Kalman Filters for Tracking Projectile-1

```
TS=1.;
ORDER=2;
PHIS=0.;
SIGTH=.01;
SIGR=100.;
VT=3000.;
GAMDEG=45.;
G=32.2;
XT=0.;
YT=0.;
XTD=VT*cos(GAMDEG/57.3);
YTD=VT*sin(GAMDEG/57.3);
```

Initial conditions on projectile

```
XR=100000.;
YR=0.;
T=0.;
S=0.;
H=.001;
PHI=zeros(ORDER,ORDER);
P=zeros(ORDER,ORDER);
IDNP=eye(ORDER);
Q=zeros(ORDER,ORDER);
PHI(1,1)=1.;
PHI(1,2)=TS;
PHI(2,2)=1.;
```

Fundamental matrix for both channels

```
HMAT(1,1)=1.;
HMAT(1,2)=0.;
```

Measurement matrix for both channels

```
PHIT=PHI';
HT=HMAT';
Q(1,1)=PHIS*TS*TS*TS/3.;
Q(1,2)=PHIS*TS*TS/2.;
Q(2,1)=Q(1,2);
Q(2,2)=PHIS*TS;
```

Process noise matrix for both channels

```
P(1,1)=1000.^2;
P(2,2)=100.^2;
PY(1,1)=1000.^2;
PY(2,2)=100.^2;
```

Initial covariance matrices

```
XTH=XT+1000.;
XTDH=XTD-100.;
YTH=YT-1000.;
YTDH=YTD-100.;
```

Initial state estimates

```
count=0;
```

MATLAB Version of Two Decoupled Polynomial Linear Kalman Filters for Tracking Projectile-2

```
while YT>=0.
        XTOLD=XT;
        XTDOLD=XTD;
        YTOLD=YT;
        YTDOLD=YTD;
        XTDD=0.;
        YTDD=-G;
        XT=XT+H*XTD;
        XTD=XTD+H*XTDD;
        YT=YT+H*YTD;
        YTD=YTD+H*YTDD;
        T=T+H;
        XTDD=0.;
        YTDD=-G;
        XT=.5*(XTOLD+XT+H*XTD);
        XTD=.5*(XTDOLD+XTD+H*XTDD);
        YT=.5*(YTOLD+YT+H*YTD);
        YTD=.5*(YTDOLD+YTD+H*YTDD);
```

Second-order Runge-Kutta integration for projectile equations

```
        S=S+H;
        if S>=(TS-.00001)
                S=0.;
                THETH=atan2((YTH-YR),(XTH-XR));
                RTH=sqrt((XTH-XR)^2+(YTH-YR)^2);
                RMAT(1,1)=(cos(THETH)*SIGR)^2+(RTH*sin(THETH)*SIGTH)^2;
```

← Downrange R

```
                PHIP=PHI*P;
                PHIPPHIT=PHIP*PHIT;
                M=PHIPPHIT+Q;
                HM=HMAT*M;
                HMHT=HM*HT;
                HMHTR=HMHT+RMAT;
                HMHTRINV(1,1)=1./HMHTR(1,1);
                MHT=M*HT;
                K=MHT*HMHTRINV;
                KH=K*HMAT;
                IKH=IDNP-KH;
                P=IKH*M;
```

Riccati equations in downrange channel

```
                THETNOISE=SIGTH*randn;
                RTNOISE=SIGR*randn;
                THET=atan2((YT-YR),(XT-XR));
                RT=sqrt((XT-XR)^2+(YT-YR)^2);
                THETMEAS=THET+THETNOISE;
                RTMEAS=RT+RTNOISE;
```

Actual noisy measurements of range and angle

```
                XTMEAS=RTMEAS*cos(THETMEAS)+XR;
                RES1=XTMEAS-XTH-TS*XTDH;
                XTH=XTH+TS*XTDH+K(1,1)*RES1;
                XTDH=XTDH+K(2,1)*RES1;
```

Downrange filter

MATLAB Version of Two Decoupled Polynomial Linear Kalman Filters for Tracking Projectile-3

```
RMATY(1,1)=(sin(THETH)*SIGR)^2+(RTH*cos(THETH)*SIGTH)^2;
```

Altitude R

```
PHIPY=PHI*PY;
PHIPPHITY=PHIPY*PHIT;
MY=PHIPPHITY+Q;
HMY=HMAT*MY;
HMHTY=HMY*HT;
HMHTRY=HMHTY+RMATY;
HMHTRINVY(1,1)=1./HMHTRY(1,1);
MHTY=MY*HT;
KY=MHTY*HMHTRINVY;
KHY=KY*HMAT;
IKHY=IDNP-KHY;
PY=IKHY*MY;
```

Riccati equations in altitude channel

```
YTMEAS=RTMEAS*sin(THETMEAS)+YR;
RES2=YTMEAS-YTH-TS*YTDH+.5*TS*TS*G;
YTH=YTH+TS*YTDH-.5*TS*TS*G+KY(1,1)*RES2;
YTDH=YTDH-TS*G+KY(2,1)*RES2;
```

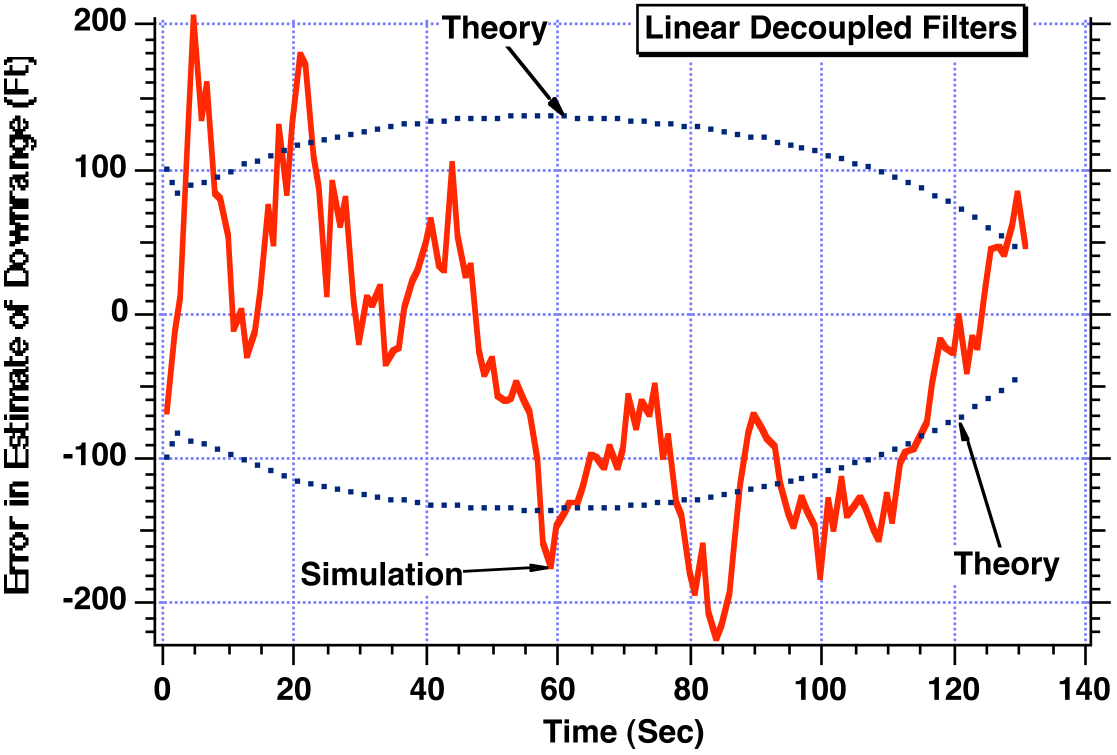
Altitude filter

```
ERRX=XT-XTH;
SP11=sqrt(P(1,1));
ERRXD=XTD-XTDH;
SP22=sqrt(P(2,2));
ERRY=YT-YTH;
SP11Y=sqrt(PY(1,1));
ERRYD=YTD-YTDH;
SP22Y=sqrt(PY(2,2));
SP11P=-SP11;
SP22P=-SP22;
SP11YP=-SP11Y;
SP22YP=-SP22Y;
count=count+1;
ArrayT(count)=T;
ArrayERRX(count)=ERRX;
ArrayERRXD(count)=ERRXD;
ArrayERRY(count)=ERRY;
ArrayERRYD(count)=ERRYD;
ArraySP11(count)=SP11;
ArraySP11P(count)=SP11P;
ArraySP22(count)=SP22;
ArraySP22P(count)=SP22P;
ArraySP11Y(count)=SP11Y;
ArraySP11YP(count)=SP11YP;
ArraySP22Y(count)=SP22Y;
ArraySP22YP(count)=SP22YP;
end
```

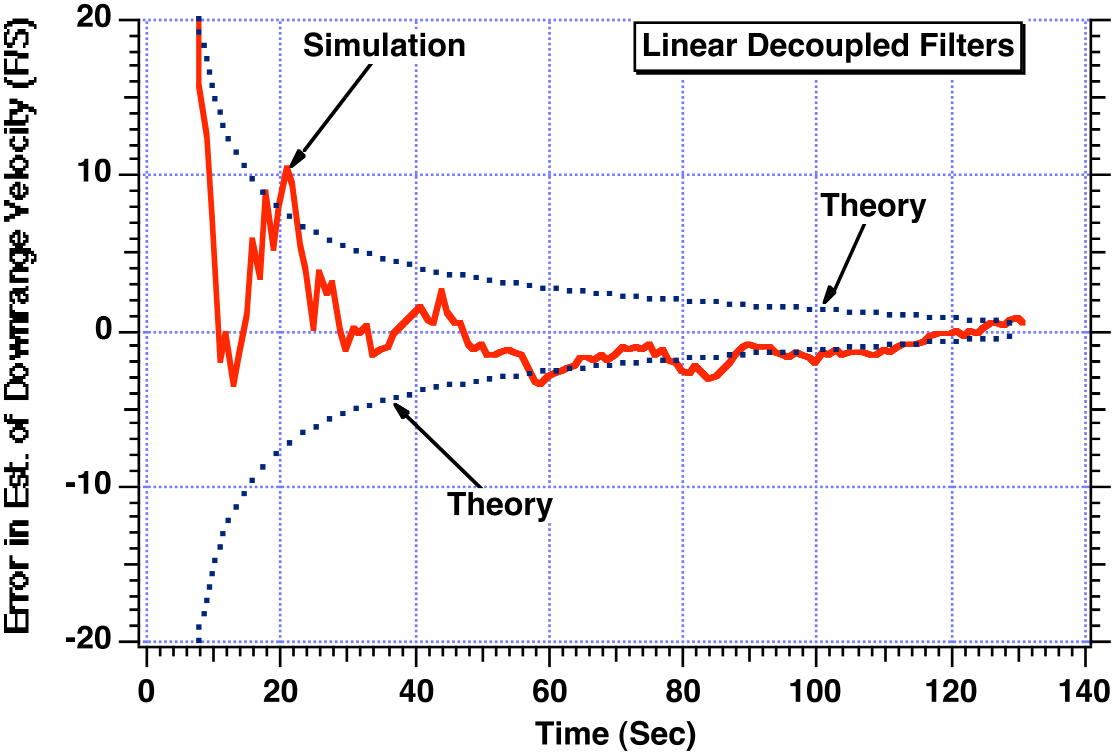
Collect data for plotting and writing to files

```
end
```

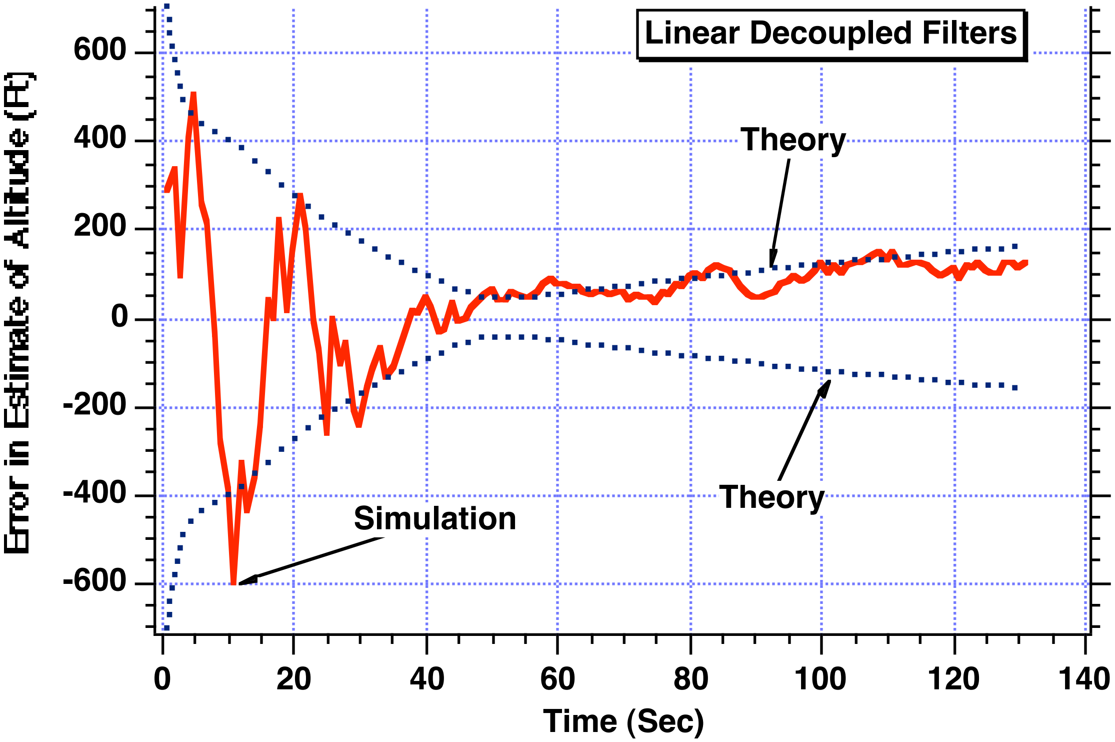
Linear Decoupled Kalman Filter Downrange Error in the Estimate of Position is Larger Than That of Extended Kalman Filter



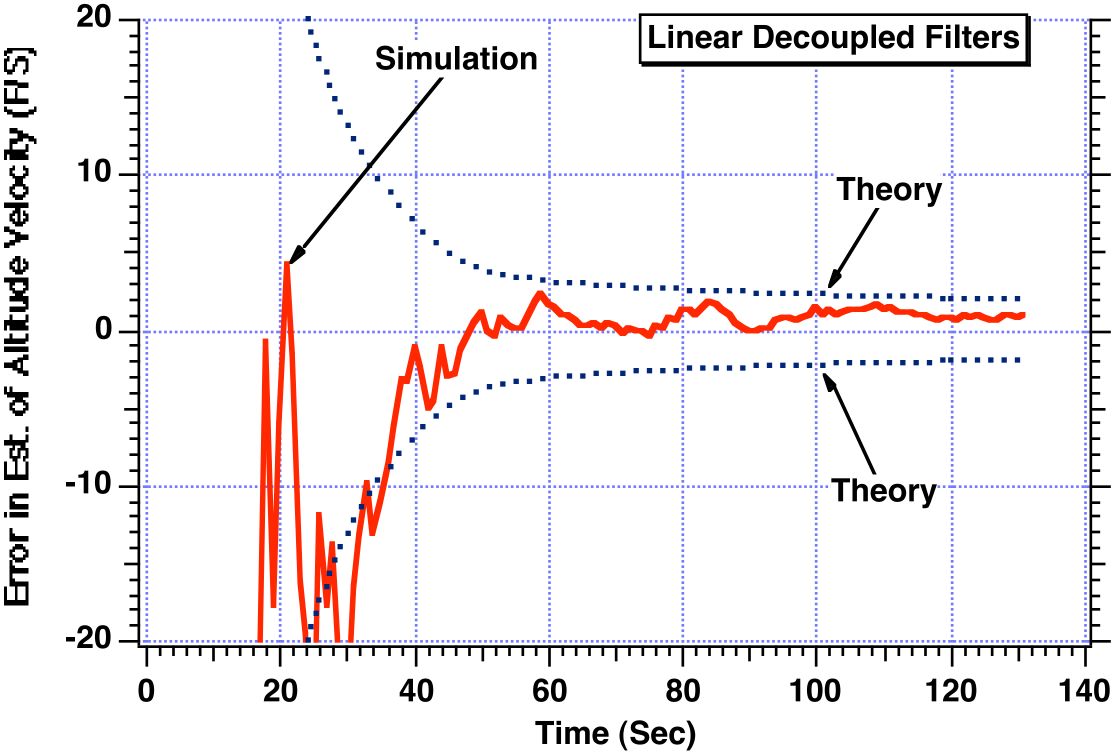
Linear Decoupled Kalman Filter Downrange Error in the Estimate of Velocity is Larger Than That of Extended Kalman Filter



Linear Decoupled Kalman Filter Altitude Error in the Estimate of Position is Larger Than That of Extended Kalman Filter



Linear Decoupled Kalman Filter Altitude Error in the Estimate of Velocity is Larger Than That of Extended Kalman Filter



Using Linear Coupled Polynomial Kalman Filters

Measurement Noise Matrix For Linear Coupled Polynomial Kalman Filter -1

In Cartesian frame model of real world is linear but measurements are nonlinear

Recall

$$x_T = r\cos\theta + x_R$$

$$y_T = r\sin\theta + y_R$$

Find total differential from calculus

$$\Delta x_T = \frac{\partial x_T}{\partial r}\Delta r + \frac{\partial x_T}{\partial \theta}\Delta\theta = \cos\theta\Delta r - r\sin\theta\Delta\theta$$

$$\Delta y_T = \frac{\partial y_T}{\partial r}\Delta r + \frac{\partial y_T}{\partial \theta}\Delta\theta = \sin\theta\Delta r + r\cos\theta\Delta\theta$$

Square both equations

$$\Delta x_T^2 = \cos^2\theta\Delta r^2 - 2r\sin\theta\cos\theta\Delta r\Delta\theta + r^2\sin^2\theta\Delta\theta^2$$

$$\Delta y_T^2 = \sin^2\theta\Delta r^2 + 2r\sin\theta\cos\theta\Delta r\Delta\theta + r^2\cos^2\theta\Delta\theta^2$$

Measurement Noise Matrix For Linear Coupled Polynomial Kalman Filter -2

We can also find

$$\Delta x_T \Delta y_T = \sin\theta\cos\theta\Delta r^2 + r\sin^2\theta\Delta r\Delta\theta + r\cos^2\theta\Delta r\Delta\theta - r^2\sin\theta\cos\theta\Delta\theta^2 \longleftarrow \text{Neglected before}$$

Taking expectations

$$E\left(\Delta x_T^2\right) = \cos^2\theta E\left(\Delta r^2\right) + r^2\sin^2\theta E\left(\Delta\theta^2\right)$$

$$E\left(\Delta y_T^2\right) = \sin^2\theta E\left(\Delta r^2\right) + r^2\cos^2\theta E\left(\Delta\theta^2\right)$$

$$E\left(\Delta x_T \Delta y_T\right) = \sin\theta\cos\theta E\left(\Delta r^2\right) - r^2\sin\theta\cos\theta E\left(\Delta\theta^2\right) \longleftarrow \text{New}$$

Therefore

$$\sigma_{x_T}^2 = \cos^2\theta\sigma_r^2 + r^2\sin^2\theta\sigma_\theta^2$$

$$\sigma_{y_T}^2 = \sin^2\theta\sigma_r^2 + r^2\cos^2\theta\sigma_\theta^2$$

$$\sigma_{x_T y_T}^2 = \sin\theta\cos\theta\sigma_r^2 - r^2\sin\theta\cos\theta\sigma_\theta^2$$

Where

$$\sigma_{x_T}^2 = E\left(\Delta x_T^2\right)$$

$$\sigma_{y_T}^2 = E\left(\Delta y_T^2\right)$$

$$\sigma_{x_T y_T}^2 = E\left(\Delta x_T \Delta y_T\right)$$

$$\sigma_r^2 = E\left(\Delta r^2\right)$$

$$\sigma_\theta^2 = E\left(\Delta\theta^2\right)$$

Or

$$\mathbf{R_k} = \begin{bmatrix} \cos^2\theta\sigma_r^2 + r^2\sin^2\theta\sigma_\theta^2 & \sin\theta\cos\theta\sigma_r^2 - r^2\sin\theta\cos\theta\sigma_\theta^2 \\ \sin\theta\cos\theta\sigma_r^2 - r^2\sin\theta\cos\theta\sigma_\theta^2 & \sin^2\theta\sigma_r^2 + r^2\cos^2\theta\sigma_\theta^2 \end{bmatrix}$$

Important Matrices For Linear Coupled Polynomial Kalman Filter-1

Measurement equation

$$\begin{bmatrix} x_T^* \\ y_T^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_T \\ \dot{x}_T \\ y_T \\ \dot{y}_T \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$

Measurement matrix

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

Model of real world

$$\begin{bmatrix} \dot{x}_T \\ \ddot{x}_T \\ \dot{y}_T \\ \ddot{y}_T \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_T \\ \dot{x}_T \\ y_T \\ \dot{y}_T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} + \begin{bmatrix} \begin{bmatrix} 0 \\ u_s \\ 0 \\ u_s \end{bmatrix} \end{bmatrix}$$

Systems dynamics matrix

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

Important Matrices For Linear Coupled Polynomial Kalman Filter-2

Fundamental matrix

$$\Phi(t) = \begin{bmatrix} 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & t \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Discrete fundamental matrix

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

Continuous process noise matrix

$$\mathbf{Q} = E(\mathbf{w}\mathbf{w}^{\mathbf{T}}) \longrightarrow \mathbf{Q}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Phi_s & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \Phi_s \end{bmatrix}$$

Discrete process noise matrix can be derived from

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

Important Matrices For Linear Coupled Polynomial Kalman Filter-3

Discrete process noise matrix

$$\mathbf{Q_k} = \begin{bmatrix} \frac{T_s^3\Phi_s}{3} & \frac{T_s^2\Phi_s}{2} & 0 & 0 \\ \frac{T_s^2\Phi_s}{2} & T_s\Phi_s & 0 & 0 \\ 0 & 0 & \frac{T_s^3\Phi_s}{3} & \frac{T_s^2\Phi_s}{2} \\ 0 & 0 & \frac{T_s^2\Phi_s}{2} & T_s\Phi_s \end{bmatrix}$$

Continuous control vector

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

Deriving discrete control vector

$$\mathbf{G_k} = \int_0^{T_s} \mathbf{\Phi G} d\tau = \int_0^{T_s} \begin{bmatrix} 0 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ -g \end{bmatrix} d\tau = \begin{bmatrix} 0 \\ 0 \\ -.5gT_s^2 \\ -gT_s \end{bmatrix}$$

Linear Coupled Polynomial Kalman Filter-1

Recall

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

Substitution yields

$$\begin{bmatrix} \hat{x}_{T_k} \\ \hat{\dot{x}}_{T_k} \\ \hat{y}_{T_k} \\ \hat{\dot{y}}_{T_k} \end{bmatrix} = \begin{bmatrix} 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{T_{k-1}} \\ \hat{\dot{x}}_{T_{k-1}} \\ \hat{y}_{T_{k-1}} \\ \hat{\dot{y}}_{T_{k-1}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -.5gT_s^2 \\ -gT_s \end{bmatrix} +$$

$$\begin{bmatrix} K_{11_k} & K_{12_k} \\ K_{21_k} & K_{22_k} \\ K_{31_k} & K_{32_k} \\ K_{41_k} & K_{42_k} \end{bmatrix} \left[\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & T_s & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_s \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{x}_{T_{k-1}} \\ \hat{\dot{x}}_{T_{k-1}} \\ \hat{y}_{T_{k-1}} \\ \hat{\dot{y}}_{T_{k-1}} \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -.5gT_s^2 \\ -gT_s \end{bmatrix} \right]$$

Linear Coupled Polynomial Kalman Filter-2

Multiplying out terms

$$RES_1 = x^*_{T_k} - \hat{x}_{T_{k-1}} - T_s\hat{\dot{x}}_{T_{k-1}}$$

$$RES_2 = y^*_{T_k} - \hat{y}_{T_{k-1}} - T_s\hat{\dot{y}}_{T_{k-1}} + .5gT_s^2$$

$$\hat{x}_{T_k} = \hat{x}_{T_{k-1}} + T_s\hat{\dot{x}}_{T_{k-1}} + K_{1\,1_k}RES_1 + K_{1\,2_k}RES_2$$

$$\hat{\dot{x}}_{T_k} = \hat{\dot{x}}_{T_{k-1}} + K_{2\,1_k}RES_1 + K_{2\,2_k}RES_2$$

$$\hat{y}_{T_k} = \hat{y}_{T_{k-1}} + T_s\hat{\dot{y}}_{T_{k-1}} -.5gT_s^2 + K_{3\,1_k}RES_1 + K_{3\,2_k}RES_2$$

$$\hat{\dot{y}}_{T_k} = \hat{\dot{y}}_{T_{k-1}} -gT_s + K_{4\,1_k}RES_1 + K_{4\,2_k}RES_2$$

MATLAB Version of Coupled Polynomial Linear Kalman Filter for Tracking Projectile-1

```
TS=1.;
ORDER=4;
PHIS=0.;
SIGTH=.01;
SIGR=100.;
VT=3000.;
GAMDEG=45.;
G=32.2;
XT=0.;
YT=0.;
XTD=VT*cos(GAMDEG/57.3);
YTD=VT*sin(GAMDEG/57.3);
```

Initial conditions on projectile

```
XR=100000.;
YR=0.;
T=0.;
S=0.;
H=.001;
PHI=zeros(ORDER,ORDER);
P=zeros(ORDER,ORDER);
IDNP=eye(ORDER);
Q=zeros(ORDER,ORDER);
PHI(1,1)=1.;
PHI(1,2)=TS;
PHI(2,2)=1.;
PHI(3,3)=1.;
PHI(3,4)=TS;
PHI(4,4)=1.;
```

Fundamental matrix

```
HMAT(1,1)=1.;
HMAT(1,2)=0.;
HMAT(1,3)=0.;
HMAT(1,4)=0.;
HMAT(2,1)=0.;
HMAT(2,2)=0.;
HMAT(2,3)=1.;
HMAT(2,4)=0.;
```

Measurement matrix

```
PHIT=PHI';
HT=HMAT';
Q(1,1)=PHIS*TS*TS*TS/3.;
Q(1,2)=PHIS*TS*TS/2.;
Q(2,1)=Q(1,2);
Q(2,2)=PHIS*TS;
Q(3,3)=PHIS*TS*TS*TS/3.;
Q(3,4)=PHIS*TS*TS/2.;
Q(4,3)=Q(3,4);
Q(4,4)=PHIS*TS;
```

Process noise matrix

MATLAB Version of Coupled Polynomial Linear Kalman Filter for Tracking Projectile-2

```
P(1,1)=1000.^2;
P(2,2)=100.^2;
P(3,3)=1000.^2;
P(4,4)=100.^2;
```

Initial covariance matrix

```
XTH=XT+1000.;
XTDH=XTD-100.;
YTH=YT-1000.;
YTDH=YTD+100.;
```

Initial filter state estimates

```
count=0;
while YT>=0.
        XTOLD=XT;
        XTDOLD=XTD;
        YTOLD=YT;
        YTDOLD=YTD;
        XTDD=0.;
        YTDD=-G;
        XT=XT+H*XTD;
        XTD=XTD+H*XTDD;
        YT=YT+H*YTD;
        YTD=YTD+H*YTDD;
        T=T+H;
        XTDD=0.;
        YTDD=-G;
        XT=.5*(XTOLD+XT+H*XTD);
        XTD=.5*(XTDOLD+XTD+H*XTDD);
        YT=.5*(YTOLD+YT+H*YTD);
        YTD=.5*(YTDOLD+YTD+H*YTDD);
```

Second-order Runge-Kutta integration for projectile equations

```
        S=S+H;
        if S>=(TS-.00001)
                S=0.;
                THETH=atan2((YTH-YR),(XTH-XR));
                RTH=sqrt((XTH-XR)^2+(YTH-YR)^2);
                RMAT(1,1)=(cos(THETH)*SIGR)^2+(RTH*sin(THETH)*SIGTH)^2;
                RMAT(2,2)=(sin(THETH)*SIGR)^2+(RTH*cos(THETH)*SIGTH)^2;
                RMAT(1,2)=sin(THETH)*cos(THETH)*(SIGR^2-(RTH*SIGTH)^2);
```

R matrix

```
                RMAT(2,1)=RMAT(1,2);
                PHIP=PHI*P;
                PHIPPHIT=PHIP*PHIT;
                M=PHIPPHIT+Q;
                HM=HMAT*M;
                HMHT=HM*HT;
                HMHTR=HMHT+RMAT;
                HMHTRINV=inv(HMHTR);
                MHT=M*HT;
                MHT=M*HT;
                K=MHT*HMHTRINV;
```

Riccati equations

MATLAB Version of Coupled Polynomial Linear Kalman Filter for Tracking Projectile-3

```
KH=K*HMAT;
IKH=IDNP-KH;
P=IKH*M;
```

```
THETNOISE=SIGTH*randn;
RTNOISE=SIGR*randn;
THET=atan2((YT-YR),(XT-XR));
RT=sqrt((XT-XR)^2+(YT-YR)^2);
THETMEAS=THET+THETNOISE;
RTMEAS=RT+RTNOISE;
```

Noisy radar measurements

```
XTMEAS=RTMEAS*cos(THETMEAS)+XR;
YTMEAS=RTMEAS*sin(THETMEAS)+YR;
```

Pseudomeasurements

```
RES1=XTMEAS-XTH-TS*XTDH;
RES2=YTMEAS-YTH-TS*YTDH+.5*TS*TS*G;
XTH=XTH+TS*XTDH+K(1,1)*RES1+K(1,2)*RES2;
XTDH=XTDH+K(2,1)*RES1+K(2,2)*RES2;
YTH=YTH+TS*YTDH-.5*TS*TS*G+K(3,1)*RES1+K(3,2)*RES2;
YTDH=YTDH-TS*G+K(4,1)*RES1+K(4,2)*RES2;
```

Filter

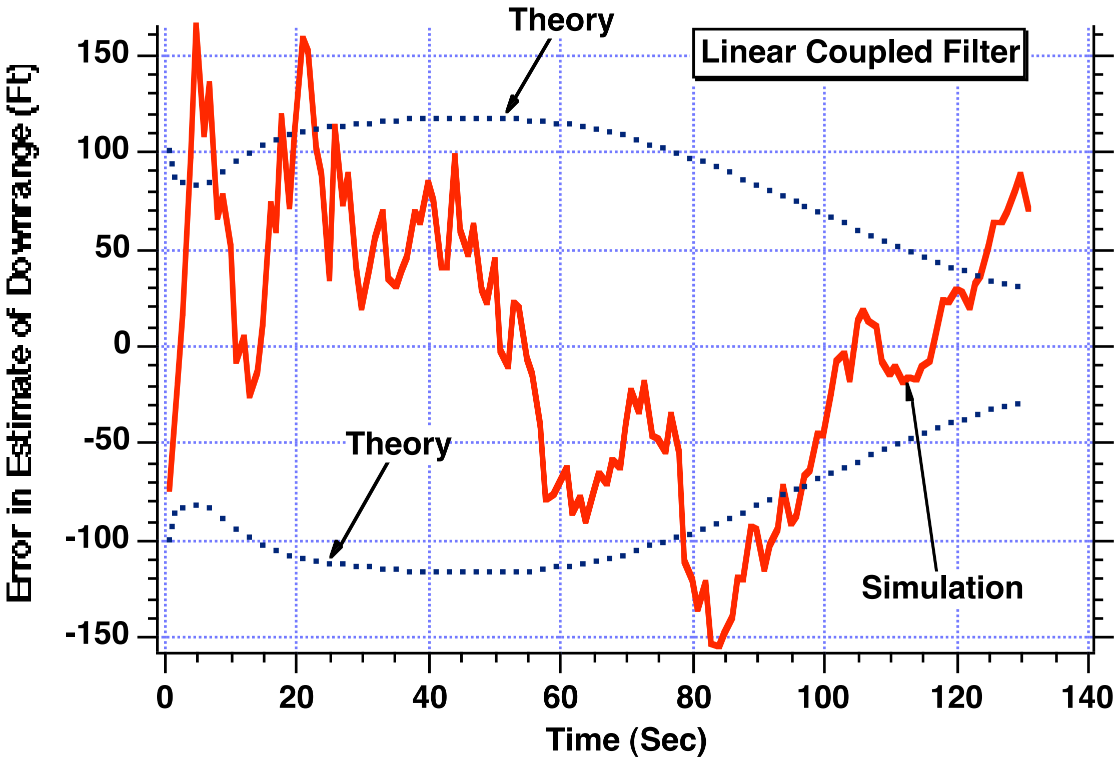
```
ERRX=XT-XTH;
SP11=sqrt(P(1,1));
ERRXD=XTD-XTDH;
SP22=sqrt(P(2,2));
ERRY=YT-YTH;
SP33=sqrt(P(3,3));
ERRYD=YTD-YTDH;
SP44=sqrt(P(4,4));
SP11P=-SP11;
SP22P=-SP22;
SP33P=-SP33;
SP44P=-SP44;
```

```
count=count+1;
ArrayT(count)=T;
ArrayERRX(count)=ERRX;
ArrayERRXD(count)=ERRXD;
ArrayERRY(count)=ERRY;
ArrayERRYD(count)=ERRYD;
ArraySP11(count)=SP11;
ArraySP11P(count)=SP11P;
ArraySP22(count)=SP22;
ArraySP22P(count)=SP22P;
ArraySP33(count)=SP33;
ArraySP33P(count)=SP33P;
ArraySP44(count)=SP44;
ArraySP44P(count)=SP44P;
end
```

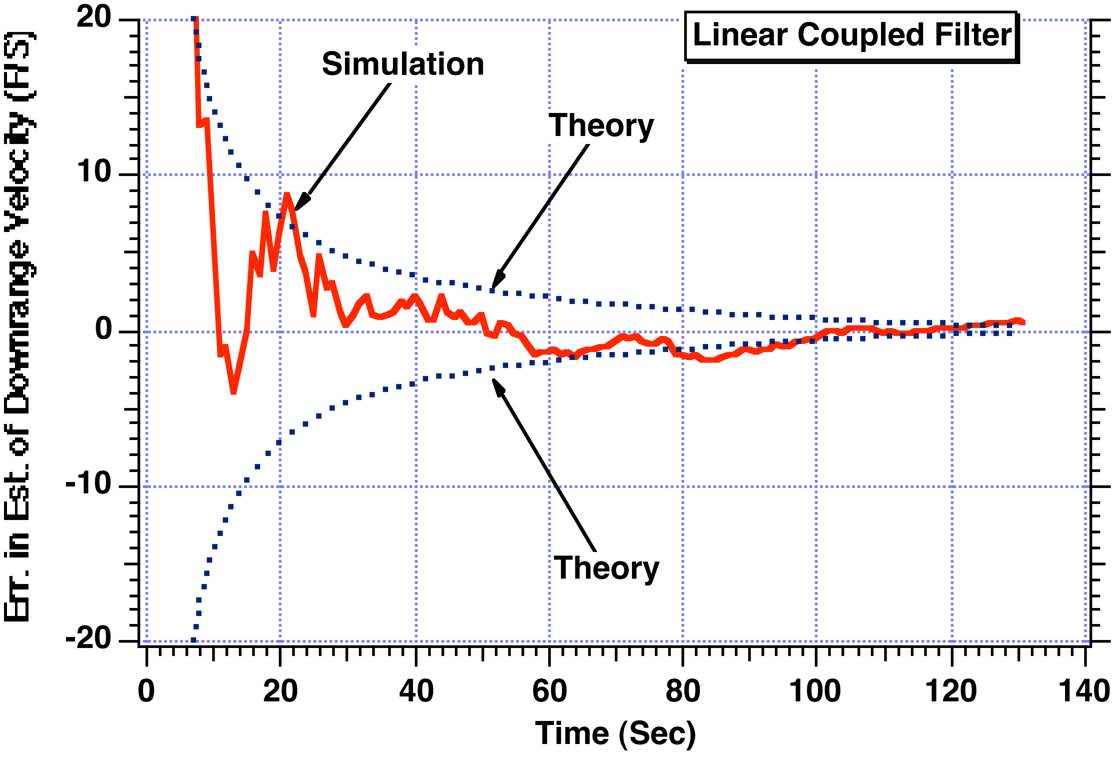
Saving data for plotting and writing to files

```
end
```

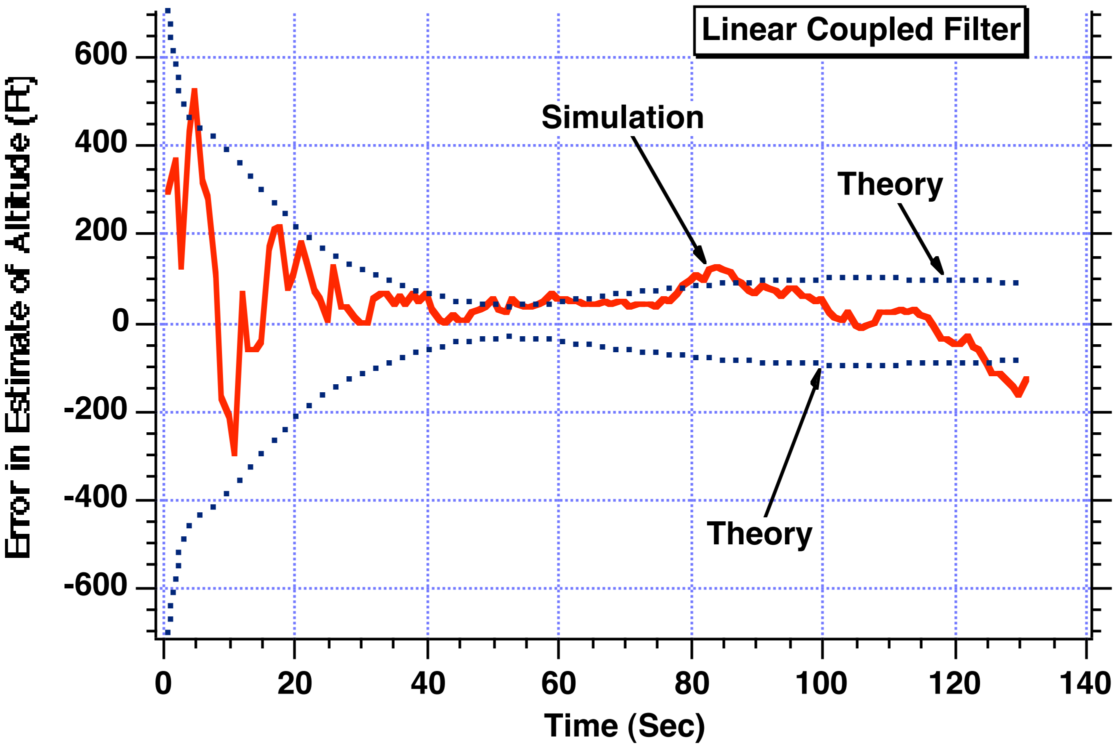
Error in Estimate of Downrange is the Same for Both the Linear Coupled and Extended Kalman Filters



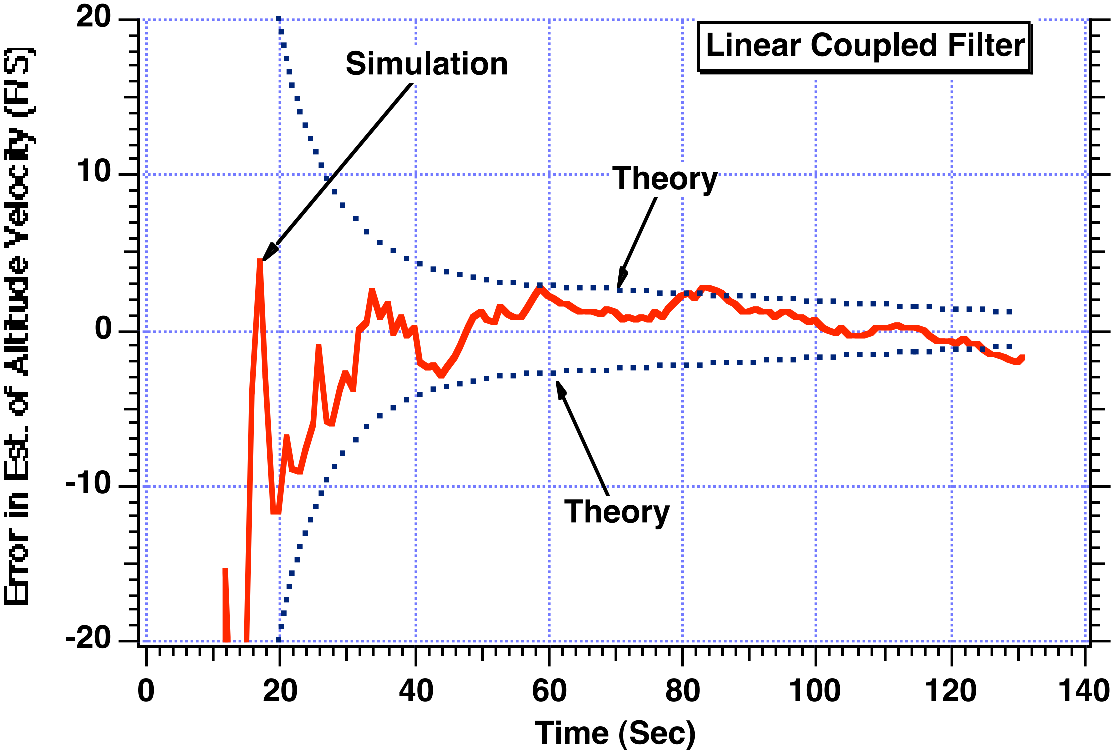
Error in Estimate of Downrange Velocity is the Same for Both the Linear Coupled and Extended Kalman Filters



Error in Estimate of Altitude is the Same for Both the Linear Coupled and Extended Kalman Filters



Error in Estimate of Altitude Velocity is the Same for Both the Linear Coupled and Extended Kalman Filters



Robustness Comparison of Extended and Linear Coupled Kalman Filters

We Will Conduct Experiments With Initialization Errors

Initial filter state estimates

$$\begin{bmatrix} \hat{x}_T(0) \\ \hat{\dot{x}}_T(0) \\ \hat{y}_T(0) \\ \hat{\dot{y}}_T(0) \end{bmatrix} = \begin{bmatrix} x_T(0) \\ \dot{x}_T(0) \\ y_T(0) \\ \dot{y}_T(0) \end{bmatrix} + \begin{bmatrix} 1000 \\ -100 \\ -1000 \\ 100 \end{bmatrix}$$

Initial covariance matrix

$$\mathbf{P}_0 = \begin{bmatrix} 1000^2 & 0 & 0 & 0 \\ 0 & 100^2 & 0 & 0 \\ 0 & 0 & 1000^2 & 0 \\ 0 & 0 & 0 & 100^2 \end{bmatrix}$$

Until now

Let us start by doubling nominal errors

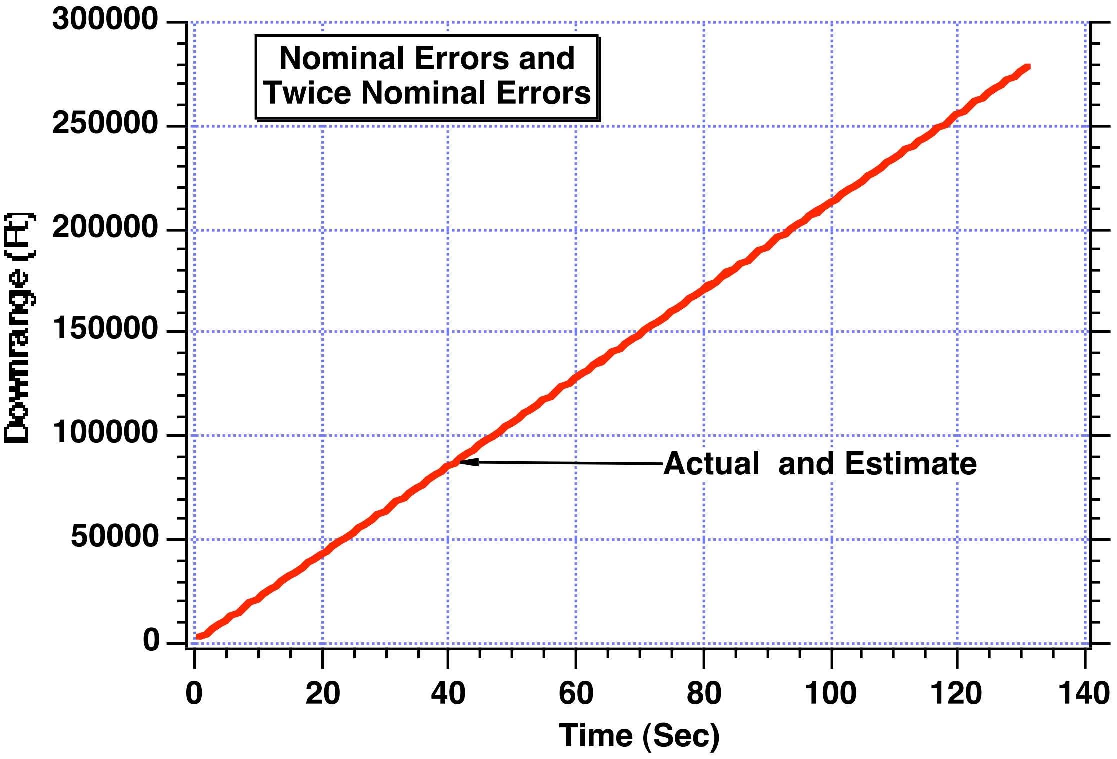
$$\begin{bmatrix} \hat{x}_T(0) \\ \hat{\dot{x}}_T(0) \\ \hat{y}_T(0) \\ \hat{\dot{y}}_T(0) \end{bmatrix} = \begin{bmatrix} x_T(0) \\ \dot{x}_T(0) \\ y_T(0) \\ \dot{y}_T(0) \end{bmatrix} + \begin{bmatrix} 2000 \\ -200 \\ -2000 \\ 200 \end{bmatrix}$$

and

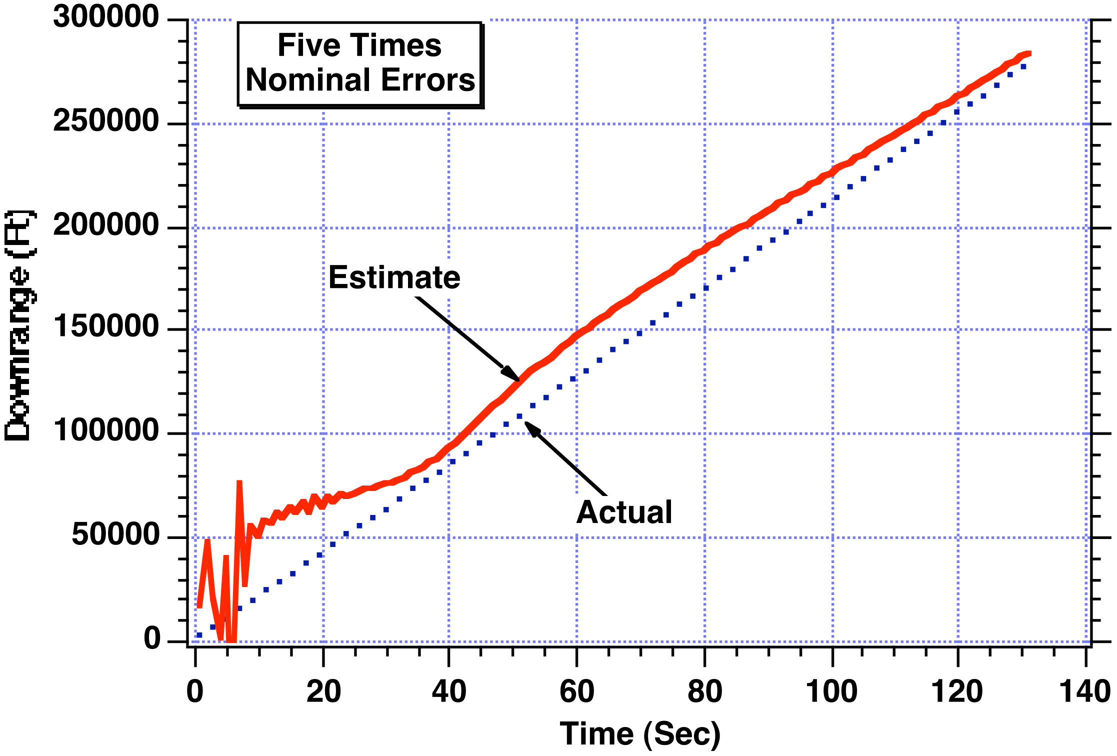
$$\mathbf{P}_0 = \begin{bmatrix} 2000^2 & 0 & 0 & 0 \\ 0 & 200^2 & 0 & 0 \\ 0 & 0 & 2000^2 & 0 \\ 0 & 0 & 0 & 200^2 \end{bmatrix}$$

Extended Kalman Filter Sensitivity to Initialization Errors

Extended Kalman Filter Appears to Yield Good Estimates Even When Initialization Errors are Twice as Large as Nominal



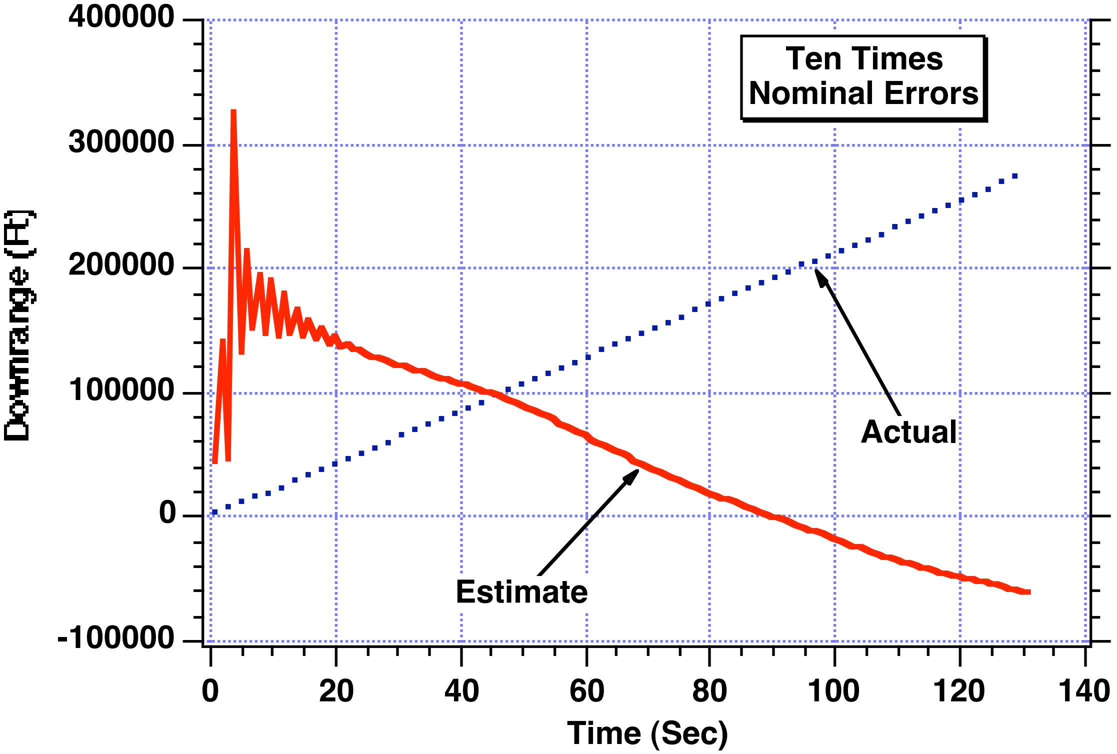
Estimates From Extended Kalman Filter Degrade Severely When Initialization Errors are Five Times Larger Than Nominal



$$\begin{bmatrix} \hat{x}_T(0) \\ \hat{\dot{x}}_T(0) \\ \hat{y}_T(0) \\ \hat{\dot{y}}_T(0) \end{bmatrix} = \begin{bmatrix} x_T(0) \\ \dot{x}_T(0) \\ y_T(0) \\ \dot{y}_T(0) \end{bmatrix} + \begin{bmatrix} 5000 \\ -500 \\ -5000 \\ 500 \end{bmatrix}$$

$$\mathbf{P}_0 = \begin{bmatrix} 5000^2 & 0 & 0 & 0 \\ 0 & 500^2 & 0 & 0 \\ 0 & 0 & 5000^2 & 0 \\ 0 & 0 & 0 & 500^2 \end{bmatrix}$$

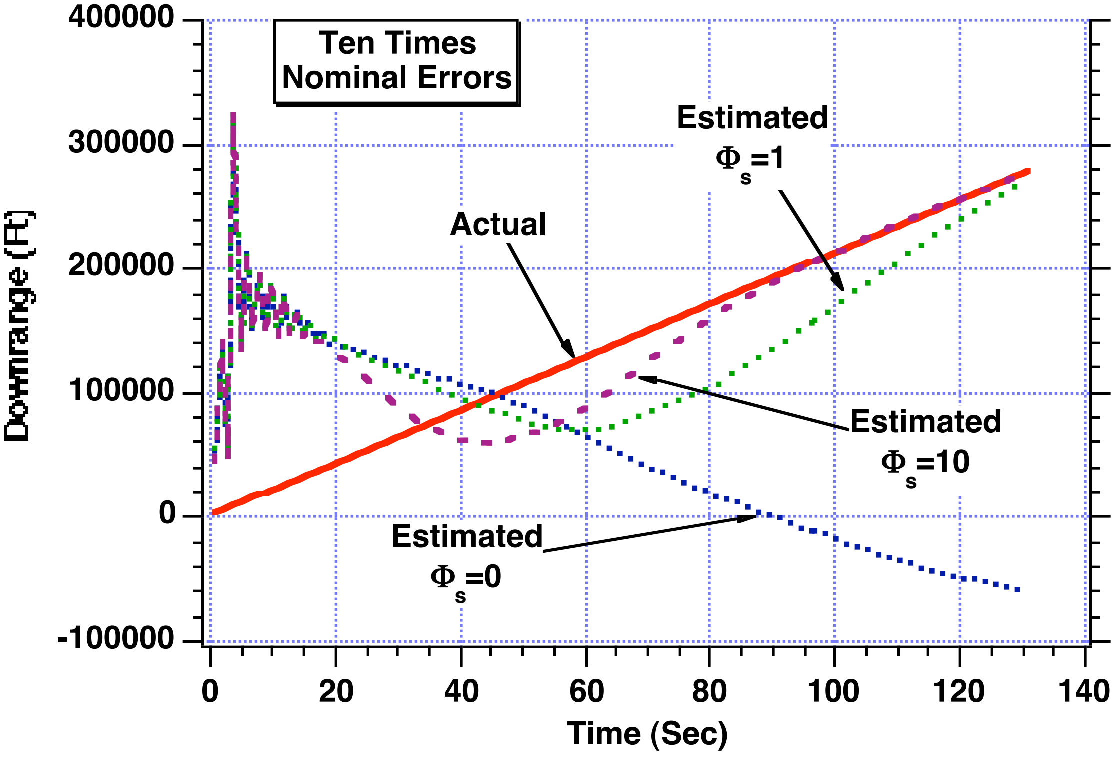
Estimates From Extended Kalman Filter are Worthless When Initialization Errors are Ten Times Larger Than Nominal



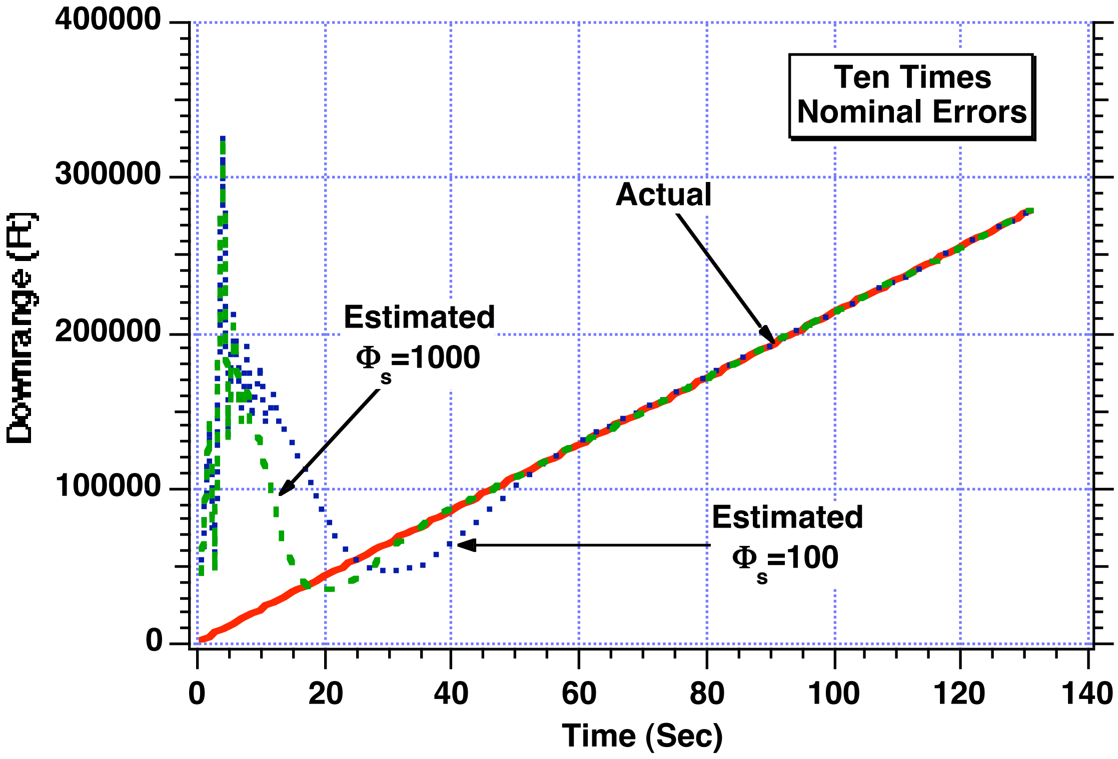
$$\begin{bmatrix} \hat{x}_T(0) \\ \hat{\dot{x}}_T(0) \\ \hat{y}_T(0) \\ \hat{\dot{y}}_T(0) \end{bmatrix} = \begin{bmatrix} x_T(0) \\ \dot{x}_T(0) \\ y_T(0) \\ \dot{y}_T(0) \end{bmatrix} + \begin{bmatrix} 10000 \\ -1000 \\ -10000 \\ 1000 \end{bmatrix}$$

$$\mathbf{P}_0 = \begin{bmatrix} 10000^2 & 0 & 0 & 0 \\ 0 & 1000^2 & 0 & 0 \\ 0 & 0 & 10000^2 & 0 \\ 0 & 0 & 0 & 1000^2 \end{bmatrix}$$

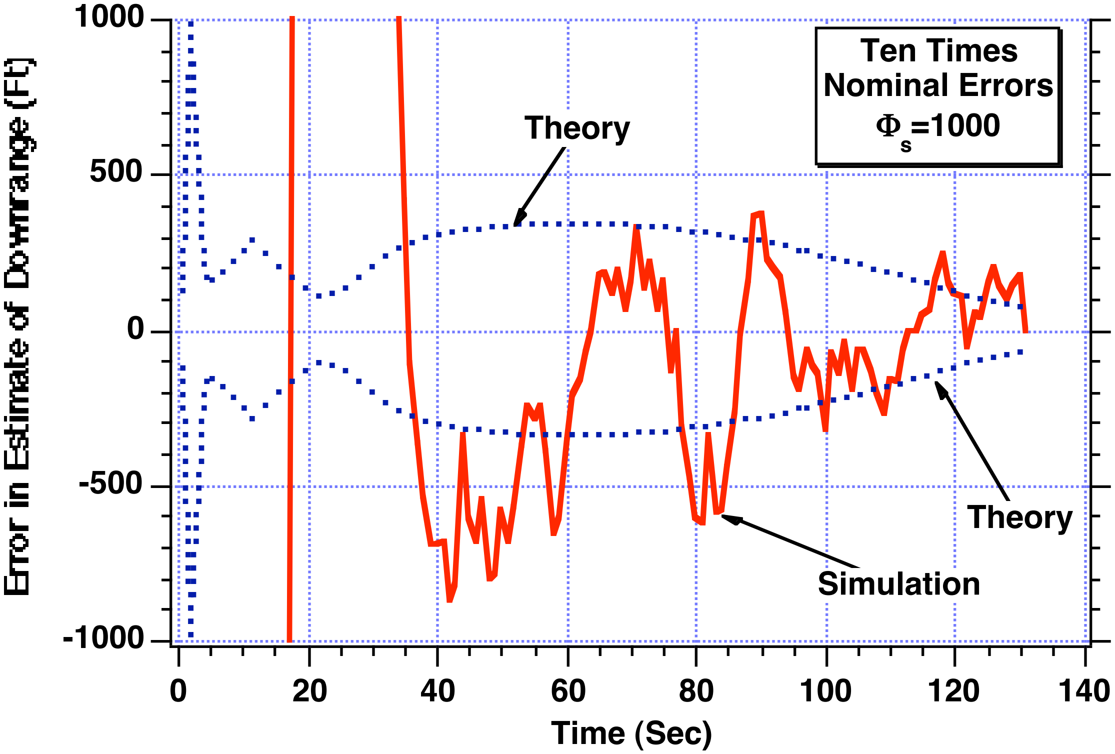
Addition of Process Noise Enables Extended Kalman Filter to Better Estimate Downrange in Presence of Large Initialization Errors



Making Process Noise Larger Further Improves Extended Kalman Filter's Ability to Estimate Downrange

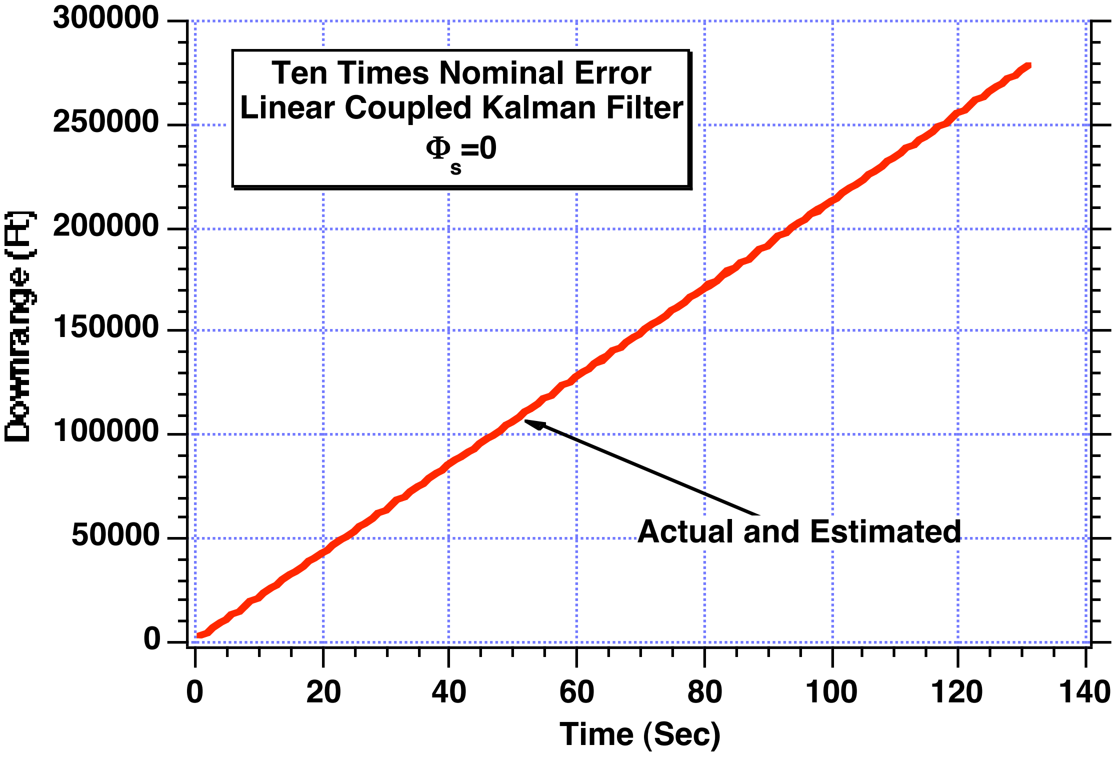


After an Initial Transient Period Simulation Results Agree With Covariance Matrix Predictions

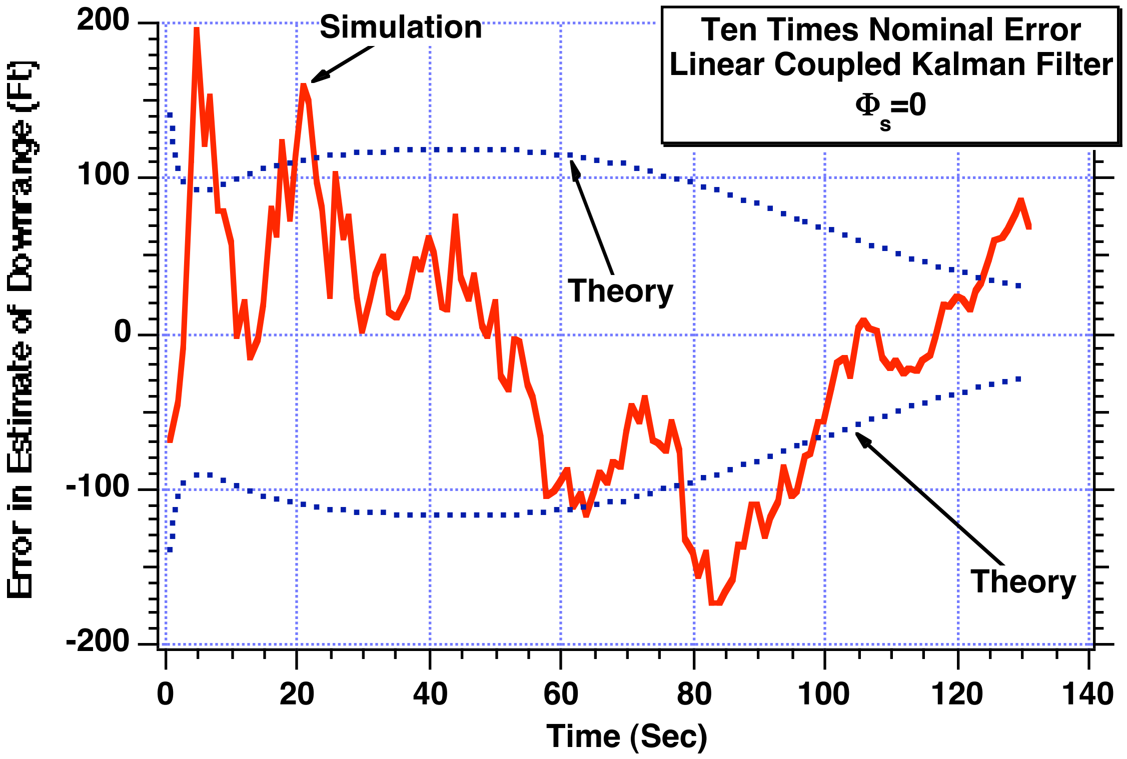


Linear Coupled Polynomial Kalman Filter Sensitivity to Initialization Errors

Linear Coupled Polynomial Kalman Filter Does Not Appear to be Sensitive to Large Initialization Errors

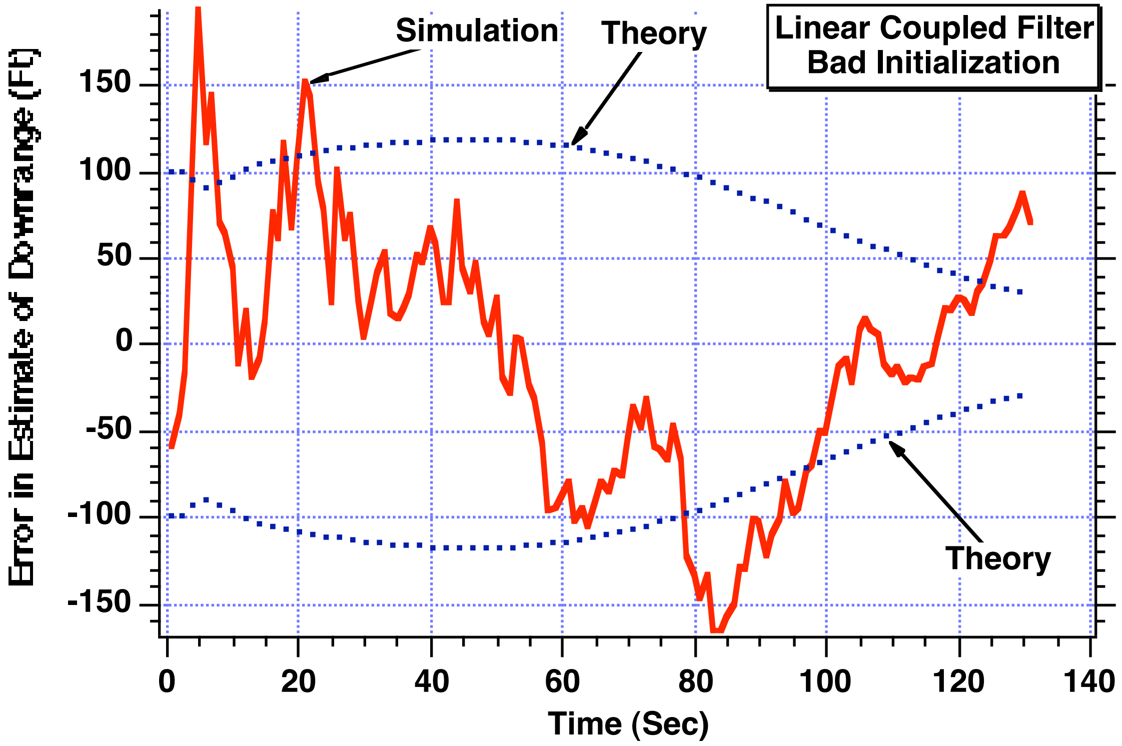


Linear Coupled Polynomial Kalman Filter Appears to be Working Correctly in Presence of Large Initialization Errors



*Same performance as when initialization errors are small

Linear Coupled Kalman Filter Performance Appears to be Independent of Initialization Errors



$$\begin{bmatrix} \hat{x}_T(0) \\ \hat{\dot{x}}_T(0) \\ \hat{y}_T(0) \\ \hat{\dot{y}}_T(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \mathbf{P}_0 = \begin{bmatrix} \infty & 0 & 0 & 0 \\ 0 & \infty & 0 & 0 \\ 0 & 0 & \infty & 0 \\ 0 & 0 & 0 & \infty \end{bmatrix}$$

Cannon Launched Projectile Tracking Problem Summary

- For this problem there were no performance advantages in the polar coordinate system but there were computational disadvantages
- For this problem a linear coupled polynomial Kalman filter yielded similar performance
- Linear coupled polynomial Kalman filter was much less sensitive to initialization errors than extended Kalman filter