Linearized Kalman Filtering
Linearized Kalman Filtering Overview

- Theoretical equations
- Falling object revisited
  - Getting feel for sensitivities
  - Designing linearized Kalman filter
- Cannon launched projectile revisited
  - Getting feel for sensitivities
  - Designing linearized Kalman filter
Theoretical Equations
Nonlinear model of the real world
\[ \dot{x} = f(x) + w \]

Process noise matrix
\[ Q = E(w w^T) \]

Nonlinear measurement equation
\[ z = h(x) + v \]

Measurement noise matrix
\[ R = E(v v^T) \]

Suppose we have nominal trajectory information
\[ \dot{x}_{NOM} = f(x_{NOM}) \]

Error equation
\[ \Delta x = x - x_{NOM} \]
Theoretical Equations - 2

Linearized differential equation for error states

\[
\Delta x = \left. \frac{\delta f(x)}{\delta x} \right|_{x=x} \Delta x + w
\]

Measurement error equation

\[
\Delta z = z - z_{\text{NOM}}
\]

Linearized error measurement equation

\[
\Delta z = \left. \frac{\delta h(x)}{\delta x} \right|_{x=x} \Delta x + v
\]

Systems dynamics and measurement matrices

\[
P = \left. \frac{\delta f(x)}{\delta x} \right|_{x=x_{\text{NOM}}}
\]

\[
H = \left. \frac{\delta h(x)}{\delta x} \right|_{x=x_{\text{NOM}}}
\]
Theoretical Equations - 3

Discrete measurement equation

$$\Delta z_k = H \Delta x_k + v_k$$

Discrete fundamental matrix

$$\Phi_k = I + FT_s + \frac{F^2 T_s^2}{2!} + \frac{F^3 T_s^3}{3!} + ...$$

Sometimes

$$\Phi_k = I + FT_s$$

Still use Riccati equations to compute filter gains

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

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

$$P_k = (I - K_k H)M_k$$

Filter

$$\Delta \hat{x}_k = \Phi \Delta \hat{x}_{k-1} + K_k (\Delta z_k - H \Phi \Delta \hat{x}_{k-1})$$

To get state estimate

$$\hat{x}_k = \Delta \hat{x}_k + x_{NOM}$$
Falling Object Revisited
Radar Tracking Object With Unknown Drag

6000 Ft/Sec

Drag
(Unknown)

g
(Known)

200,000 Ft
Equations Representing Real World

Acceleration acting on object

\[ \ddot{x} = \text{Drag} - g = \frac{Q_p g}{\beta} - g \]

\[ Q_p = .5 \rho x^2 \]

\[ \rho = .0034 e^{-x/22000} \]

Rewriting differential equation

\[ \ddot{x} = \frac{Q_p g}{\beta} - g = \frac{.5 \rho x^2}{\beta} - g = \frac{.0034 g e^{-x/22000} x^2}{2\beta} - g \]
FORTRAN Simulation for Actual and Nominal Trajectories of Falling Object-1

Initial conditions for actual trajectory

Initial conditions for nominal trajectory

Integrating actual and nominal differential equations with second-order Runge-Kutta technique

Fundamentals of Kalman Filtering: A Practical Approach
FORTRAN Simulation for Actual and Nominal Trajectories of Falling Object-2

IF(S >= (TS-.00001)) THEN
  S = 0.
  DELXH = X-XNOM
  DELXDH = XD-XDNOM
  DELBETAH = BETA-BETANOM
  XH = XNOM + DELXH
  XDH = XDNOM + DELXDH
  BETAH = BETANOM + DELBETAH
  WRITE(9,*), DELXH, DELXDH, DELBETAH, XH, XD, XDH
  WRITE(1,*), DELXH, DELXDH, DELBETAH, XH, XD, XDH
ENDIF
END DO
PAUSE
CLOSE(1)
END
Object Slows Up Significantly as it Races Towards the Ground (Nominal Case)
Differences Between the Actual and Nominal Ballistic Coefficient Can Yield Significant Differences in Altitude
Differences Between the Actual and Nominal Ballistic Coefficient Can Yield Significant Differences in Velocity
Developing A Linearized Kalman Filter For Falling Object Problem
Developing A Linearized Kalman Filter-1

Error states

\[
\Delta x = \begin{bmatrix} \Delta x \\ \Delta \dot{x} \\ \Delta \beta \end{bmatrix} = \begin{bmatrix} x - x_{\text{NOM}} \\ \dot{x} - \dot{x}_{\text{NOM}} \\ \beta - \beta_{\text{NOM}} \end{bmatrix}
\]

Model of real world

\[
\ddot{x} = 0.0034ge^{-x/22000x^2} - g
\]
\[
\beta = u_s
\]

Error model equation

\[
\begin{bmatrix} \Delta \ddot{x} \\ \Delta \dddot{x} \\ \Delta \beta \end{bmatrix} = \begin{bmatrix} \frac{\partial \ddot{x}}{\partial x} & \frac{\partial \ddot{x}}{\partial \dot{x}} & \frac{\partial \ddot{x}}{\partial \beta} \\ \frac{\partial \dddot{x}}{\partial x} & \frac{\partial \dddot{x}}{\partial \dot{x}} & \frac{\partial \dddot{x}}{\partial \beta} \\ \frac{\partial \beta}{\partial x} & \frac{\partial \beta}{\partial \dot{x}} & \frac{\partial \beta}{\partial \beta} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \dot{x} \\ \Delta \beta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ u_s \end{bmatrix}
\]

\[\text{x = x}_{\text{NOM}}\]
Developing A Linearized Kalman Filter-2

**Systems dynamics matrix**

\[
F = \begin{bmatrix}
\frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial \beta} \\
\frac{\partial \ddot{x}}{\partial x} & \frac{\partial \ddot{x}}{\partial x} & \frac{\partial \ddot{x}}{\partial \beta} \\
\frac{\partial \beta}{\partial x} & \frac{\partial \beta}{\partial x} & \frac{\partial \beta}{\partial \beta}
\end{bmatrix}_{x=\text{NOM}}
\]

**Continuous process noise matrix**

\[
Q = E(ww^T) \quad \rightarrow \quad Q(t) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \Phi_s \end{bmatrix}
\]

**Evaluate first row of partial derivatives**

\[
\dot{x} = \ddot{x} \quad \rightarrow \quad \frac{\partial \ddot{x}}{\partial x} = 0, \quad \frac{\partial \ddot{x}}{\partial \beta} = 1, \quad \frac{\partial \ddot{x}}{\partial \beta} = 0
\]
Developing A Linearized Kalman Filter-3

Evaluate second row of partial derivatives

\[ \dot{x} = \frac{0.0034e^{-x/22000} \cdot x^2}{2\beta} - g \]

\[ \frac{\partial \dot{x}}{\partial x} = \frac{-0.0034e^{-x/22000} \cdot g}{2\beta(22000)} = \frac{-\rho g x^2}{44000\beta} \]

\[ \frac{\partial \dot{x}}{\partial \beta} = \frac{2 \cdot 0.0034e^{-x/22000} \cdot x g}{2\beta} = \frac{\rho gx}{\beta} \]

\[ \frac{\partial \dot{x}}{\partial \dot{x}} = \frac{-0.0034e^{-x/22000} \cdot g}{2\beta^2} = \frac{-\rho g x^2}{2\beta^2} \]

Evaluate third row of partial derivatives

\[ \ddot{\beta} = u_s \]

\[ \frac{\partial \ddot{\beta}}{\partial x} = 0 \]

\[ \frac{\partial \ddot{\beta}}{\partial \dot{x}} = 0 \]

\[ \frac{\partial \ddot{\beta}}{\partial \beta} = 0 \]
Developing A Linearized Kalman Filter-4

Systems dynamics matrix is evaluated along nominal trajectory

\[ F = \begin{bmatrix} 0 & 1 & 0 \\ \frac{-\rho_{\text{nom}}x_{\text{nom}}^2}{44000\beta_{\text{nom}}} & \rho_{\text{nom}}x_{\text{nom}} & -\frac{\rho_{\text{nom}}x_{\text{nom}}^2}{2\beta_{\text{nom}}} \\ 0 & 0 & 0 \end{bmatrix} \]

where \( \rho_{\text{nom}} = .0034 e^{-x_{\text{nom}}/22000} \)

Two term Taylor series approximation for fundamental matrix

\[ \Phi(t) = I + Ft \]

If we rewrite systems dynamics matrix as

\[ F(t) = \begin{bmatrix} 0 & 1 & 0 \\ f_{21} & f_{22} & f_{23} \\ 0 & 0 & 0 \end{bmatrix} \]

where

\[ f_{21} = \frac{-\rho_{\text{nom}}x_{\text{nom}}^2}{44000\beta_{\text{nom}}} \]

\[ f_{22} = \frac{\rho_{\text{nom}}x_{\text{nom}}^2}{\beta_{\text{nom}}} \]

\[ f_{23} = \frac{-\rho_{\text{nom}}x_{\text{nom}}^2}{2\beta_{\text{nom}}} \]
Developing A Linearized Kalman Filter-5

Then

$$\Phi(t) = I + Ft = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ f_{21} & f_{22} & f_{23} \\ 0 & 0 & 0 \end{bmatrix}t = \begin{bmatrix} 1 & t & 0 \\ f_{21}t & 1+f_{22}t & f_{23}t \\ 0 & 0 & 1 \end{bmatrix}$$

And discrete fundamental matrix becomes

$$\Phi_k = \begin{bmatrix} 1 & T_s & 0 \\ f_{21}T_s & 1+f_{22}T_s & f_{23}T_s \\ 0 & 0 & 1 \end{bmatrix}$$

Error measurement equation

$$\Delta x^* = \begin{bmatrix} \Delta x \\ \Delta \dot{x} \\ \Delta \beta \end{bmatrix} + v$$

Therefore measurement matrix is

$$H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$
Developing A Linearized Kalman Filter-6

Discrete measurement noise matrix is a scalar

\[
R_k = E(v_k v_k^T) \quad \rightarrow \quad R_k = \sigma_v^2
\]

Continuous process noise matrix

\[
Q = E(ww^T) \quad \rightarrow \quad Q(t) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_w^2 \end{bmatrix}
\]

Discrete process noise matrix

\[
Q_k = \int_0^{T_s} \Phi(\tau)Q\Phi^T(\tau)d\tau
\]

\[
Q_k = \Phi_s \int_0^{T_s} \begin{bmatrix} 1 & \tau & 0 \\ f_{21}\tau & 1+f_{22}\tau & f_{23}\tau \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & f_{21}\tau & 0 \\ f_{22}\tau & 1+f_{22}\tau & f_{23}\tau \\ f_{23}\tau & 0 & 0 \end{bmatrix} d\tau
\]

\[
Q_k = \Phi_s \int_0^{T_s} \begin{bmatrix} 0 & 0 & 0 \\ 0 & f_{23}^2\tau^2 & f_{23}\tau \\ 0 & f_{23}\tau & 1 \end{bmatrix} d\tau
\]

\[
Q_k = \Phi_s \begin{bmatrix} 0 & 0 & 0 \\ 0 & f_{23}T_s^3/3 & f_{23}T_s^2/2 \\ 0 & f_{23}T_s^2/2 & T_s \end{bmatrix}
\]
Linearized Kalman filtering equation

\[
\begin{bmatrix}
\Delta \hat{x}_k \\
\Delta \hat{x}_k \\
\Delta \beta_k
\end{bmatrix} = \Phi
\begin{bmatrix}
\Delta \hat{x}_{k-1} \\
\Delta \hat{x}_{k-1} \\
\Delta \beta_{k-1}
\end{bmatrix} + 
\begin{bmatrix}
K_{1k} \\
K_{2k} \\
K_{3k}
\end{bmatrix}
(\Delta x_k^* - H \Phi
\begin{bmatrix}
\Delta \hat{x}_{k-1} \\
\Delta \hat{x}_{k-1} \\
\Delta \beta_{k-1}
\end{bmatrix})
\]
MATLAB Simulation of Linearized Kalman Filter-1

Initial actual and nominal states

\[
\begin{align*}
X &= 200000.; \\
XD &= -6000.; \\
BETA &= 500.; \\
XNOM &= 200000.; \\
XDNOM &= -6000.; \\
BETANOM &= 500.; \\
ORDER &= 3; \\
TS &= .1; \\
TF &= 30.; \\
Q33 &= 300.*300./TF; \\
T &= 0.; \\
S &= 0.; \\
H &= .001; \\
SIGNoise &= 25.; \\
RMAT(1,1) &= SIGNoise^2; \\
XH(1,1) &= X-XNOM; \\
XH(2,1) &= XD-XDNOM; \\
XH(3,1) &= -300.; \\
PHI &= \text{zeros}(ORDER,ORDER); \\
P &= \text{zeros}(ORDER,ORDER); \\
IDNP &= \text{eye}(ORDER); \\
Q &= \text{zeros}(ORDER,ORDER); \\
P(1,1) &= SIGNoise*SIGNoise; \\
P(2,2) &= 20000.; \\
P(3,3) &= 300.\,^2; \\
HMAT &= \text{zeros}(1,ORDER); \\
HMAT(1,1) &= 1.; \\
HT &= HMAT^*; \\
\end{align*}
\]

Initial filter error state estimates

Initial covariance matrix
while X>=0.

XOLD=X;
XDOLD=XD;
XNOMOLD=XNOM;
XDNONMOLD=XDNONM;
XDD=0.0034*32.2*XD*XD*exp(-X/22000.)/(2.*BETA)-32.2;
XDNONM=0.0034*32.2*XDNONM*XDNONM*exp(-XNOM/22000.)/(2.*BETANOM)-32.2;
X=X+H*XD;
XD=XD+H*XD;
XNOM=XNOM+H*XDNONM;
XDNONM=XDNONM+H*XDNONM;
T=T+H;
XDD=0.0034*32.2*XD*XD*exp(-X/22000.)/(2.*BETA)-32.2;
XDNONM=0.0034*32.2*XDNONM*XDNONM*exp(-XNOM/22000.)/(2.*BETANOM)-32.2;
X=0.5*(XOLD+X+H*XD);
XD=0.5*(XDOLD+XD+H*XD);
XNOM=0.5*(XNOMOLD+XNOM+H*XDNONM);
XDNONM=0.5*(XDNONMOLD+XDNONM+H*XDNONM);
S=S+H;
if S>=(TS-.00001)
    S=0.;
end
RHONOM=0.0034*exp(-XNOM/22000.);
F21=-32.2*RHONOM*XDNONM*XDNONM/(2.*22000.*BETANOM);
F22=RHONOM*32.2*XDNONM/BETANOM;
F23=-RHONOM*32.2*XDNONM*XDNONM/(2.*BETANOM*BETANOM);
PHI(1,1)=1.;
PHI(1,2)=TS;
PHI(2,1)=F21*TS;
PHI(2,2)=1.+F22*TS;
PHI(2,3)=F23*TS;
PHI(3,3)=1.;
Q(2,2)=F23*F23*Q33*TS*TS/3.;
Q(2,3)=F23*Q33*TS*TS/2.;
Q(3,2)=Q(2,3);
Q(3,3)=Q33*TS;
PHIT=PHI';

Numerical integration of actual and nominal equations of falling object

Fundamental matrix

Process noise matrix
MATLAB Simulation of Linearized Kalman Filter-3

```
PHIP=PHI*P;
PHIPPHIT=PHIP*PHIT;
M=PHIPPHIT+Q;
HM=HMAT*M;
HMHT=HM*HT;
HMTR=HMHT+RMAT;
HMTRINV(1,1)=1./HMTR(1,1);
MHT=M*HT;
K=MHT*HMTRINV;
KH=K*HMAT;
IKH=IDNP-KH;
P=IKH*M;
XNOISE=SIGNoise*randn;
DELX=X-XNOISE;
MEAS(1,1)=DELX+XNOISE;
PHIXH=PHI*XH;
HPHIXH=HMAT*PHIXH;
RES=MEAS-HPHIXH;
KRES=K*RES;
XH=PHIXH+KRES;
DELXH=XH(1,1);
DELBETHA=XH(2,1);
DELBETHAH=XH(3,1);
XHH=XH+DELXH;
XDH=XH+DELXDH;
BETAH=BETHA+DELBETHAH;
ERRX=X-XHH;
SP11=sqrt(P(1,1));
ERRXD=XD-XDH;
SP22=sqrt(P(2,2));
ERRBETHA=BETA-BETHA;
SP33=sqrt(P(3,3));
SP11P=-SP11;
SP22P=-SP22;
SP33P=-SP33;
count=count+1;
ArrayT(count)=T;
ArrayBETA(count)=BETA;
ArrayBETHA(count)=BETHA;
ArrayERRBETHA(count)=ERRBETHA;
ArraySP33(count)=SP33;
ArraySP33P(count)=SP33P;
end
```

Riccati equations

Filter in terms of error states

Convert estimated error states to estimated states

Actual and theoretical errors in estimates

Save data in arrays for plotting and writing to files
Linearized Kalman Filter is able to accurately estimate the ballistic coefficient after only 10 seconds.

*Of course if actual and nominal ballistic coefficients were equal we would not have to estimate ballistic coefficient.*
Linearized Kalman Filter Appears to be Working Correctly
Linearized Kalman Filter Accurately Estimates Ballistic Coefficient When There is a Slight Error in the Nominal Trajectory

![Graph showing the comparison between actual and estimated ballistic coefficients over time. The actual ballistic coefficient starts at 500 Lb/Ft² and the nominal at 600 Lb/Ft². The graph indicates that the estimate closely follows the actual values, with only minor deviations.]
Error in Estimate of Ballistic Coefficient is Between Theoretical Error Bounds When There is a Slight Error in the Nominal Trajectory
Linearized Kalman Filter Underestimates Ballistic Coefficient When There is 60% Error in the Nominal Ballistic Coefficient

Actual

Estimate

\[ \beta_{\text{ACT}} = 500 \text{ Lb/Ft}^2 \]

\[ \beta_{\text{NOM}} = 800 \text{ Lb/Ft}^2 \]
Error in Estimate of Ballistic Coefficient is Not Between Theoretical Error Bounds With 60% Error in the Nominal Ballistic Coefficient
There is Significant Steady-State error in the Linearized Kalman Filter’s Estimate of Ballistic Coefficient When There is a 120% Error

\[ \beta_{\text{ACT}} = 500 \text{ Lb/Ft}^2 \]

\[ \beta_{\text{NOM}} = 1100 \text{ Lb/Ft}^2 \]
With 120% Error, the Errors in the Estimates are Completely Outside the Theoretical Error Bounds

Simulation

Theory

\[ \beta_{\text{ACT}} = 500 \text{ Lb/ft}^2 \]

\[ \beta_{\text{NOM}} = 1100 \text{ Lb/ft}^2 \]
Cannon Launched Projectile Revisited
Radar Tracking Cannon Launched Projectile

**Acceleration on projectile**

\[
\begin{align*}
\dot{x}_T &= 0 \\
\dot{y}_T &= -g
\end{align*}
\]

**Angle and range from radar to projectile**

\[
\begin{align*}
\theta &= \tan^{-1}\left[\frac{y_T - y_R}{x_T - x_R}\right] \\
r &= \sqrt{(x_T - x_R)^2 + (y_T - y_R)^2}
\end{align*}
\]
**FORTRAN Simulation for Actual and Nominal Projectile Trajectories-1**

```
IMPLICIT REAL*8(A-H,O-Z)
TS=1.
VT=3000.
VTERR=0.
GAMDEG=45.
GAMDEGERR=0.
VTNOM=VT+VTERR
GAMDEGNOM=GAMDEG+GAMDEGERR
G=32.2
XT=0.
YT=0.
XTD=VT*COS(GAMDEG/57.3)
YTD=VT*SIN(GAMDEG/57.3)
XTNOM=XT
YTNOM=YT
XTDNOM=VTNOM*COS(GAMDEGNOM/57.3)
YTDNOM=VTNOM*SIN(GAMDEGNOM/57.3)
XR=100000.
YR=0.
OPEN(1,STATUS='UNKNOWN',FILE='DATFIL')
T=0.
S=0.
H=.001
WHILE(YT>=0.)
  XTOld=XT
  XTDold=XTD
  YTOld=YT
  YTDold=YTD
  XTNOMold=XTNOM
  XTDNOMold=XTDNOM
  YTNOMold=YTNOM
  YTDNOMold=YTDNOM
  XTDd=0.
  YTDd=-G
  XTDNOMd=0.
  YTDNOMd=-G
  XT=XT+H*XTD
  XTD=XTD+H*XTDd
  YT=YT+H*YTD
  YTD=YTD+H*YTDd
```

Actual and nominal initial conditions

Integration of actual and nominal differential equations using the second-order Runge-Kutta technique
FORTRAN Simulation for Actual and Nominal Projectile Trajectories-2

XTNOM = XTNOM + H * XTDNOM
XTDNOM = XTDNOM + H * XTDNOM
YTNOM = YTNOM + H * YTDNOM
YTDNOM = YTDNOM + H * YTDDNOM
T = T + H
XTDD = 0.
YTDD = G
XTDDNOM = 0.
YTDDNOM = G
XT = 5 * (XTOLD + XT + H * XTD)
XTD = 5 * (XTDOLD + XTD + H * XTDD)
YT = 5 * (YTDOLD + YT + H * YTD)
YTDD = 5 * (YTDDOLD + YTDD + H * YTDD)
XTNOM = 5 * (XTNOMOLD + XTNOM + H * XTDNOM)
XTDNOM = 5 * (XTDNOMOLD + XTDNOM + H * XTDDNOM)
YTNOM = 5 * (YTNOMOLD + YTNOOM + H * YTDDNOM)
YTDDNOM = 5 * (YTDDNOMOLD + YTDDNOM + H * YTDDNOM)
S = S + H
IF(S >= (TS - 0.00001)) THEN
  S = 0.
  RTNOM = SQRT((XTNOM - XR)**2 + (YTNOM - YR)**2)
  THETNOM = ATAN2((YTNOM - YR), (XTNOM - XR))
  RT = SQRT((XT - XR)**2 + (YT - YR)**2)
  THET = ATAN2((YT - YR), (XT - XR))
  DELTHET = 57.3 * (THET - THETNOM)
  DELRT = RT - RTNOM
  DELXT = XT - XTNOM
  DELXTD = XTD - XTDNOM
  DELYT = YT - YTNOM
  DELYTD = YTD - YTDDNOM
  WRITE(9, *) T, DELXT, DELXTD, DELYT, DELYTD, DELTHET,
  DELRT
  WRITE(1, *) T, DELXT, DELXTD, DELYT, DELYTD, DELTHET,
  DELRT
ENDIF
END DO
PAUSE
CLOSE(1)
END
Differences Between the Actual and Nominal Initial Projectile Velocities Can Yield Significant Downrange Differences

\[ V_{\text{TACT}} = 3000 \text{ Ft/Sec} \]

\[ V_{\text{NOM}} = 3000 \text{ Ft/Sec} \]

\[ V_{\text{NOM}} = 3050 \text{ Ft/Sec} \]

\[ V_{\text{NOM}} = 3100 \text{ Ft/Sec} \]

\[ V_{\text{NOM}} = 3200 \text{ Ft/Sec} \]
Differences Between the Actual and Nominal Initial Projectile Velocities Can Yield Significant Angle Differences

\[ V_{\text{TACT}} = 3000 \text{ Ft/Sec} \]
\[ V_{\text{NOM}} = 3000 \text{ Ft/Sec} \]
\[ V_{\text{NOM}} = 3100 \text{ Ft/Sec} \]
\[ V_{\text{NOM}} = 3200 \text{ Ft/Sec} \]
\[ V_{\text{NOM}} = 3300 \text{ Ft/Sec} \]
\[ V_{\text{NOM}} = 3400 \text{ Ft/Sec} \]
Differences Between the Actual and Nominal Initial Projectile Velocities Can Yield Significant Range Differences

Differences in Initial Velocities:
- $V_{ACT} = 3000 \text{ Ft/Sec}$
- $V_{NOM} = 3000 \text{ Ft/Sec}$
- $V_{NOM} = 3050 \text{ Ft/Sec}$
- $V_{NOM} = 3100 \text{ Ft/Sec}$
- $V_{NOM} = 3200 \text{ Ft/Sec}$

Graph showing the error in range (in feet) over time (in seconds) for different nominal velocities.
Linearized Cartesian Kalman Filter
Developing a Linearized Cartesian Kalman Filter-1

Error states for filter

\[
\Delta x = \begin{bmatrix}
\Delta x_T \\
\Delta \dot{x}_T \\
\Delta y_T \\
\Delta \dot{y}_T
\end{bmatrix}
= \begin{bmatrix}
x_T-x_{TNOM} \\
\dot{x}_T-\dot{x}_{TNOM} \\
y_T-y_{TNOM} \\
\dot{y}_T-\dot{y}_{TNOM}
\end{bmatrix}
\]

Model of real world

\[
\begin{bmatrix}
\Delta x_T \\
\Delta \dot{x}_T \\
\Delta y_T \\
\Delta \dot{y}_T
\end{bmatrix}
= \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
\Delta x_T \\
\Delta \dot{x}_T \\
\Delta y_T \\
\Delta \dot{y}_T
\end{bmatrix}
+ \begin{bmatrix}
0 \\
u_s \\
0 \\
u_s
\end{bmatrix}
\]

Systems dynamics matrix

\[
F = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

Taylor series approximation for fundamental matrix

\[
\Phi(t) = I + Ft + \frac{F^2 t^2}{2!} + \frac{F^3 t^3}{3!} + ...
\]
Developing a Linearized Cartesian Kalman Filter-2

Since

\[
F^2 = \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix}\begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix} = \begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

Two term approximation is exact

\[
\Phi(t) = I + Ft = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}t + \begin{bmatrix}
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0
\end{bmatrix}t
\]

\[
\Phi_k = \begin{bmatrix}
0 & T_s & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & T_s \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

Discrete fundamental matrix

Error measurement equation

\[
\begin{bmatrix}
\Delta \theta^* \\
\Delta r^*
\end{bmatrix} = \begin{bmatrix}
\frac{\partial \theta}{\partial x_T} & \frac{\partial \theta}{\partial x_T} & \frac{\partial \theta}{\partial y_T} & \frac{\partial \theta}{\partial y_T}
\end{bmatrix} \begin{bmatrix}
\Delta x_T \\
\Delta x_T \\
\Delta y_T \\
\Delta y_T
\end{bmatrix} + \begin{bmatrix}
v_\theta \\
v_r
\end{bmatrix}
\]
Developing a Linearized Cartesian Kalman Filter-3

First row of partials

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

\[ \frac{\partial \theta}{\partial x_T} = \frac{1}{1 + \left( \frac{y_T - y_R}{x_T - x_R} \right)^2} \left( \frac{x_T - x_R}{(x_T - x_R)^2} - \frac{y_T - y_R}{r^2} \right) \]

\[ \frac{\partial \theta}{\partial y_T} = 0 \]

\[ \frac{\partial \theta}{\partial y_R} = \frac{1}{1 + \left( \frac{y_T - y_R}{x_T - x_R} \right)^2} \left( \frac{x_T - x_R}{(x_T - x_R)^2} - \frac{y_T - y_R}{r^2} \right) \]

Second row of partials

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

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

\[ \frac{\partial r}{\partial x_R} = 0 \]

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

\[ \frac{\partial r}{\partial y_R} = 0 \]
Developing a Linearized Cartesian Kalman Filter

Since measurement matrix is

\[ H = \begin{bmatrix}
\frac{\partial \theta}{\partial x_T} & \frac{\partial \theta}{\partial y_T} & \frac{\partial \theta}{\partial \phi} & \frac{\partial \theta}{\partial \psi} \\
\frac{\partial r}{\partial x_T} & \frac{\partial r}{\partial y_T} & \frac{\partial r}{\partial \phi} & \frac{\partial r}{\partial \psi}
\end{bmatrix} \]

We get

\[ H = \begin{bmatrix}
-(y_T - y_R) & 0 & \frac{x_T - x_R}{r^2} & 0 \\
\frac{x_T - x_R}{r} & 0 & \frac{y_T - y_R}{r^2} & 0
\end{bmatrix} \]

Evaluate discrete matrix along nominal trajectory

\[ H_k = \begin{bmatrix}
-(y_{T,NOM} - y_R) & 0 & \frac{x_{T,NOM} - x_R}{r_{NOM}^2} & 0 \\
\frac{x_{T,NOM} - x_R}{r_{NOM}} & 0 & \frac{y_{T,NOM} - y_R}{r_{NOM}^2} & 0
\end{bmatrix} \]
Developing a Linearized Cartesian Kalman Filter-5

Discrete measurement noise matrix

\[ R_k = E(v_k v_k^T) \]

Continuous process noise matrix

\[ Q = E(w w^T) \]

General formula for discrete process noise matrix

\[ Q_k = \int_0^{T_s} \Phi(\tau) Q \Phi^T(\tau) d\tau \]

Substitution yields

\[ Q_k = \int_0^{T_s} \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \Phi_s & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \Phi_s & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ \tau & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \tau & 1 \end{bmatrix} d\tau \]
Developing a Linearized Cartesian Kalman Filter-6

After some algebra we get

\[
Q_k = \int_0^{T_k} \begin{bmatrix}
\tau^2 \Phi_s & \tau \Phi_s & 0 & 0 \\
\tau \Phi_s & \Phi_s & 0 & 0 \\
0 & 0 & \tau^2 \Phi_s & \tau \Phi_s \\
0 & 0 & \tau \Phi_s & \Phi_s
\end{bmatrix} d\tau
\]

Integration yields

\[
Q_k = \begin{bmatrix}
\frac{T_k^3 \Phi_s}{3} & \frac{T_k^2 \Phi_s}{2} & 0 & 0 \\
\frac{T_k^2 \Phi_s}{2} & T_k \Phi_s & 0 & 0 \\
0 & 0 & \frac{T_k^3 \Phi_s}{3} & \frac{T_k^2 \Phi_s}{2} \\
0 & 0 & \frac{T_k^2 \Phi_s}{2} & T_k \Phi_s
\end{bmatrix}
\]

Linearized Kalman filter
MATLAB Linearized Kalman Filter for Tracking Cannon Launched Projectile-1

TS=1.; ORDER=4; PHIS=0.; SIGTH=.01; SIGR=100.; VT=3000.; VTER=0.; GAMDEG=45.; GAMDEGERR=0.; VTNOM=VT+VTER; GAMDEGNOM=GAMDEG+GAMDEGERR; G=32.2; XT=0.; YT=0.;

XTD=VT*cos(GAMDEG/57.3); YTD=VT*sin(GAMDEG/57.3); XT=XT; YTNOM=XT; YTNOM=YT;

XTDNOM=VTNOM*cos(GAMDEGNOM/57.3); YTDNOM=VTNOM*sin(GAMDEGNOM/57.3); XR=100000.; YR=0.; T=0.; S=0.; H=.001;

XH(1,1)=XT-XTNOM; XH(2,1)=XTD-XTDNOM; XH(3,1)=YT-YTNOM; XH(4,1)=YTD-YTDNOM;

PHI=zeros(ORDER,ORDER); P=zeros(ORDER,ORDER); IDNP=eye(ORDER); Q=zeros(ORDER,ORDER); TS2=TS*TS; TS3=TS2*TS;
MATLAB Linearized Kalman Filter for Tracking
Cannon Launched Projectile-2

\[
Q(1,1) = \Phi IS^3/3; \\
Q(1,2) = \Phi IS^2/2; \\
Q(2,1) = Q(1,2); \\
Q(2,2) = \Phi IS; \\
Q(3,3) = Q(1,1); \\
Q(3,4) = Q(1,2); \\
Q(4,3) = Q(3,4); \\
Q(4,4) = Q(2,2); \\
\Phi I(1,1) = 1.; \\
\Phi I(1,2) = TS; \\
\Phi I(2,2) = 1.; \\
\Phi I(3,3) = 1.; \\
\Phi I(3,4) = TS; \\
\Phi I(4,4) = 1.; \\
\Phi IT = \Phi I'; \\
R MAT(1,1) = SIGTH^2; \\
R MAT(1,2) = 0.; \\
R MAT(2,1) = 0.; \\
R MAT(2,2) = SIGTH^2; \\
P(1,1) = 1000.^2; \\
P(2,2) = 100.^2; \\
P(3,3) = 1000.^2; \\
P(4,4) = 100.^2; \\
count = 0; \\
\text{while } YT >= 0.
\]

\[
X TOLD = XT; \\
X TDOLD = XTD; \\
Y TOLD = YT; \\
Y TDOLD = YTD; \\
XT NOMOLD = XTNOM; \\
X TDNOMOLD = XTDNOM; \\
Y TNOMOLD = YTNOM; \\
Y TDNOMOLD = YTDNOM; \\
XT DD = 0.; \\
Y TD = G;
\]
MATLAB Linearized Kalman Filter for Tracking Cannon Launched Projectile-3

Integrate actual and nominal differential equations using second-order Runge-Kutta technique

Measurement matrix

Fundamentals of Kalman Filtering: A Practical Approach
MATLAB Linearized Kalman Filter for Tracking Cannon Launched Projectile-4

Riccati equations

Linearized filter in terms of error states

Convert error state estimates to state estimates

Actual and theoretical errors in estimates
MATLAB Linearized Kalman Filter for Tracking Cannon Launched Projectile-5

SP11P = SP11;
SP22P = SP22;
SP33P = SP33;
SP44P = SP44;
count = count + 1;
ArrayT(count) = T;
ArrayXT(count) = XT;
ArrayXTH(count) = XTH;
ArrayXTD(count) = XTD;
ArrayXTDH(count) = XTDH;
ArrayYT(count) = YT;
ArrayYTH(count) = YTH;
ArrayYTD(count) = YTD;
ArrayYTDH(count) = YTDH;
ArrayERRX(count) = ERRX;
ArrayERRXD(count) = ERRXD;
ArrayERRY(count) = ERY;
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
figure
plot(ArrayT, ArrayERRX, ArrayT, ArraySP11, ArrayT, ArraySP11P), grid
xlabel('Time (Sec)')
ylabel('Error in Estimate of x (Ft)')
axis([0 140 -170 170])

Save data as arrays for plotting and writing to files

Sample plot
Downrange Errors in the Estimates are Within the Theoretical Bounds for Perfect Nominal Trajectory
Downrange Velocity Errors in the Estimates are Within the Theoretical Bounds for Perfect Nominal Trajectory
Altitude Errors in the Estimates are Within the Theoretical Bounds for Perfect Nominal Trajectory
Altitude Velocity Errors in the Estimates are Within the Theoretical Bounds for Perfect Nominal Trajectory

![Graph showing error in altitude velocity estimates compared to perfect nominal trajectory and theory. The graph displays time in seconds on the x-axis and error in altitude velocity in feet per second on the y-axis. The graph includes plots for simulation and theory, with theoretical bounds indicated.]
Error in Estimate of Projectile Downrange is Between Theoretical Error Bounds When There is a 1.7% Error in the Nominal Velocity
Error in Estimate of Projectile Downrange Starts to Drift Outside Theoretical Error Bounds When There is a 3.3% Error in the Nominal Velocity

![Graph showing error in estimate of downrange over time against theoretical and simulated results. The graph illustrates how the error increases with time due to a 3.3% error in the nominal velocity.](image-url)
Error in Estimate of Projectile Downrange Has No Relationship to Theoretical Error Bounds When There is a 6.7% Error in the Nominal Velocity
Linearized Kalman Filtering
Summary

• When nominal trajectories are accurate linearized Kalman filter yields excellent estimates
  - Of course in this case filter may not be required
• When nominal trajectories are inaccurate linearized Kalman filter yields estimates that deteriorate
  - Under certain circumstances they may be worthless