Nonlinear system model for this system is given by as follows.





















Matlab code for Environment

function [stateout, meas] = Envrionment(state0, wk, m)

% Initialization

deltat = 0.1;

g=9.836;

persistent state; %variable to save states of environment

if isempty(state)

state = state0; %initialize the state using state0 from input.

end

h\_ = 2;

bet\_ = 150;

rho0\_ = 1.225;

kp\_ = 9200;

%

ry = state(2);

vy = state(4);

deltah = state(5);

deltabet = state(6);

deltarho = state(7);

deltakp = state(8);

%

h=h\_+deltah;

bet=bet\_+deltabet;

rho0=rho0\_+deltarho;

kp=kp\_+deltakp;

%

alpha = rho0\*exp(-ry/kp)\*vy\*vy/(2\*m\*bet);

% calculate the state transverse matrix. This is calculated using equation above and also used in EKF.

phi = [1 0 deltat 0 0 0 0 0; 0 1 0 deltat 0 0 0 0; 0 0 1 0 0 0 0 0; 0 -alpha\*deltat/kp 0 1+rho0\*exp(-ry/kp)\*vy/(m\*bet)\*deltat 0 -alpha/bet\*deltat alpha/rho0\*deltat ry\*alpha\*deltat/(kp\*kp);

0 0 0 0 1 0 0 0; 0 0 0 0 0 1 0 0; 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 1];

u = [0; -g\*deltat\*deltat/2; 0; -g\*deltat\*(1+rho0\*exp(-ry/kp)\*vy/(2\*m\*bet)); 0; 0; 0; 0];

%update state and mix with process noise wk which is given by input.You can set the covariance for process noise in the Simulink.

state = phi\*state + u + wk;

% Measurement matrix to be used to output measurements.

H = [1 0 0 0 0 0 0 0; 0 1 0 0 0 0 0 0];

b = [0; -h];

%calculate the measurements.

y = H\*state + b;

meas = y;

%output the updated state

stateout = state;

**Matlab code for EKF**

function [residual,xhatPreOut, xhatOut] = EXTKALMAN(meas)

persistent P;

persistent xhat

if isempty(P)

xhat = [0; 1000; 0; 0; 0; 0; 0; 0];

P = zeros(8);

P(2)=10;

P(4)=1;

end

g=9.836;

delta\_t=0.5;

m = 1;

h\_ = 2;

bet\_ = 150;

rho0\_ = 1.225;

kp\_ = 9200;

%get the states of last step to calculate the state transverse matrix.

ry = xhat(2);

vy = xhat(4);

deltah = xhat(5);

deltabet = xhat(6);

deltarho = xhat(7);

deltakp = xhat(8);

%

h=h\_+deltah;

bet=bet\_+deltabet;

rho0=rho0\_+deltarho;

kp=kp\_+deltakp;

%

alpha = rho0\*exp(-ry/kp)\*vy\*vy/(2\*m\*bet);

%

% 1. Compute Phi, Q, and R

% calculate phi from the equation above. It is same used in environment.

Phi = [1 0 delta\_t 0 0 0 0 0; 0 1 0 delta\_t 0 0 0 0; 0 0 1 0 0 0 0 0; 0 -alpha\*delta\_t/kp 0 1+rho0\*exp(-ry/kp)\*vy/(m\*bet)\*delta\_t 0 -alpha/bet\*delta\_t alpha/rho0\*delta\_t ry\*alpha\*delta\_t/(kp\*kp);

0 0 0 0 1 0 0 0; 0 0 0 0 0 1 0 0; 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 1];

Q = diag([0 10 0 1 0 0 0 0 ]);

R = diag([1 4]);

% 2. Propagate the covariance matrix:



P = Phi\*P\*Phi' + Q;

% 3. Propagate the track estimate::

u = [0; -g\*delta\_t\*delta\_t/2; 0; -g\*delta\_t\*(1+rho0\*exp(-ry/kp)\*vy/(2\*m\*bet)); 0; 0; 0; 0];

xhat = Phi\*xhat+u;

%output the prio estimates.

xhatPreOut = xhat;

% 4 a). Compute observation estimates:

H = [1 0 0 0 0 0 0 0; 0 1 0 0 0 0 0 0];

b = [0; -h];



yhat = H\*xhat + b;

% 4 c). Compute residual (Estimation Error)



residual = meas - yhat;

% 5. Compute Kalman Gain:



W = P\*H'\*inv(H\*P\*H'+ R);

%

% % 6. Update post estimates



xhat = xhat + W\*residual;

%

% % 7. Update Covariance Matrix



P = (eye(8)-W\*H)\*P\*(eye(8)-W\*H)' + W\*R\*W';

%output the post estimates.

xhatOut = xhat;

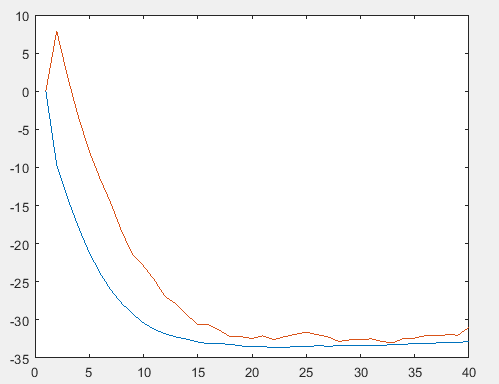


Fig1. ± standard deviation for

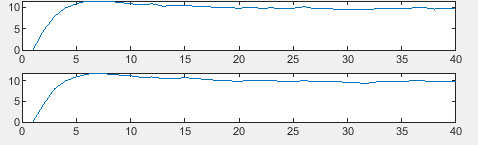


Fig2.covariance for post and prio estimates for

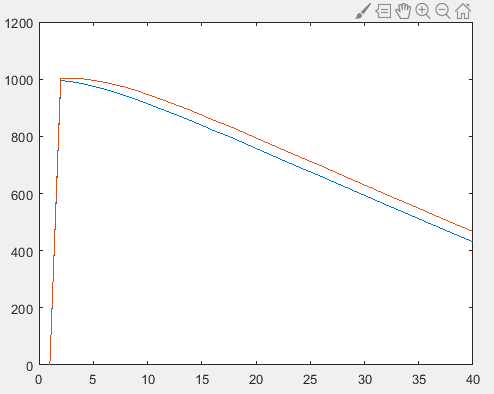


Fig3 ± standard deviation for

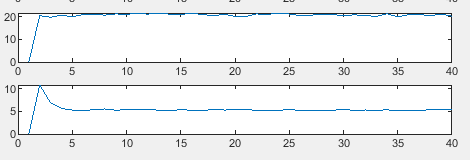


Fig4.covariance for post and prio estimates for