EECE 522 Completion Project

Arna Friend

# Introduction

This document details an extended Kalman filter, designed for an aircraft moving in a straight line with constant velocity. The path of the aircraft is modeled in x-y coordinates, but the measurement of its position is in range and bearing. The Kalman filter uses a model based on the (in this case very simple) physics of the situation to make an estimation of the state of the system at the next instant using the estimate of the previous state (resulting in a delay of one sampling period and a need to initialize the filter with values before any measurements are taken). The prediction of the state is used to make a prediction of the observation, whose model is non-linear in this case. The difference between the actual measurement and the predicted measurement (the innovation) corresponds to the portion of the data uncorrelated with the previous measurement.

The Kalman filter uses the concept of Bayesian estimation, in which the expected PDF of the random variable(s) under estimation is refined with each realization of the RV(s). In the Kalman filter, the dynamics and observation models provide the expected behavior, and the innovation provides the means to predict, with increasing accuracy over time, how the system is actually behaving without any noise.

is the output of the filter and represents the prediction of the filter’s state given all previous data points and is given by:

This formulation is possible because the innovation and the previous data are not correlated (a scalar multiplier is considered here for simplicity. In the actual filter, a is matrix ).

Similarly to , the formulation of is possible because is uncorrelated with .

The estimation, the observation, and the prediction all have error associated with them, which are accounted for in the following development by the covariance matrices **,** and , respectively. All sources of error are modeled as zero mean additive white Gaussian noise. The error associated with the prediction is due to physical factors not included in the model, such as wind. The error associated with the observation is due to electrical noise on sensor measurements. initializes , which indicates the accuracy of the previous estimate once the filter is running. The value of depends on confidence in the accuracy of the initial position. and are referred to as the mean square (MSE) error matrices, each corresponding to the estimate. indicates the error associated with the previous estimate conditioned on all data up to and including that point. is the error associated with the current prediction, and is formed by adding the previous prediction’s MSE (put onto the same basis as the state) to the estimated prediction noise. is the variance between the actual difference between the real state and the estimate and the Kalman gain times the innovation. Over time, these two quantities should become the same and the Kalman gain becomes constant.

# Physical Model for Prediction

The diagram at right represents the physical situation described above, where the purple line represents the predicted path of the aircraft, without un-modeled physical effects or noise in sensor measurements. The x and y positions, and velocities (modeled as constant) together comprise the state vector of the system:



The motion of the plane can be modeled by the following equations, where simply represents a small step in time.

The linear dynamic model, which provides the basis for the estimation of system states, is represented in state variable form by the following equation, where represents the state transition matrix, which predicts the position at time n from the position at time n-1. Vector represents noise due to un-modeled effects such as wind, and matrix distributes the driving noise across the states. Since the wind’s effect on the position would ultimately be a consequence of its effect on the velocity, the noise is only added to the velocities. For simplicity, in this model, the noise is distributed equally between x and y, but in reality the effects of wind would likely vary with altitude. is assumed to be normally distributed, zero mean noise.

The driving noise, , is generated for test purposes by generating a 4x1 vector of random numbers (one for each state variable) between -1 and 1. The noise on the positions is canceled by, for the reasons previously discussed. The expected value of forms the initial estimation, (indicated by the “hat”) which is the estimation of the initial state conditioned on itself of the system:

Since the noise on the positions is canceled, and the velocities are constant, the initial estimate is just the initial positions and velocities (the expected value of a constant being a constant and the expected value of the noise being zero).

# Measurement Model and Observation Model

The observation model (again applying the assumption of constant velocity), is linear and shown below (the two states besides range and bearing are speed and heading):

The measurement model transforms the variables used in the physical model into the variables that are measured ), using their physical relationships and takes the form below (for each scalar element):

Since the position measurements are taken as range and bearing, the measurement model is not linear. The following equations show the transformations between x and y and x and y velocities:

In order to be put in matrix form, must be expressed as a linear combination of x and y:

A simple way to derive a and b is to consider what and are equal to. These expressions, summed, must equal R. So:

Similarly, theta must be expressed as a linear combination of x and y:

The expression for can be rearranged using the following identity and properties:

The expressions for c and d in terms of and follow:

In terms of x and y, they are:

The equation for the observation takes the following form:

Since is a non-linear, it must be linearized on each iteration with a first order Taylor series expanded about the estimation of This results in the following linearized matrix:

# Results of Running Code and Error Analysis

Below is a graph of the filter results. I used variances of .0625, 0027,1,1,1, and 1 for range, bearing, x, y, x’ and y’ respectively. The position error is not relevant and is cancelled by the B matrix.



The most illustrative plot is of the error in x and y over time. In this plot, the progression of the system into a LTI system with zero steady state error is very apparent:



# Code: Test Procedure

I did it from the command line, but I also checked the linearized observation model by the observations generated by the direct, non-linear relations to those generated by the Linear Model.

% Test Code for the revision of a simple EKF

%% Generate Data (Both States and Obsevations

Cn = [.0625 0;0 .0027]; %Measurement Covariance Matrix: Using a variance of .0625 for range (std of .25m) and a variance of .0027 for the bearing (std of 1 degree)

Q = eye(4); % Physical Model Covariance Matrix:1 m^2 variance on the x and y positions,1 m^4/s^2 on velocity (although the position error is cancelled by B to avoid redundant error)

init\_state = [1 1 100 100]';% state variables are x,y,x',y'

N = 1000; %1000 time instants

B = [zeros(2,4);zeros(2,2) eye(2)]; % Matrix to distribute model error among the states

[states\_true,obs\_true]= state\_gen\_V3(Q,Cn,init\_state,N,B); % the states and observations are returned as columns

% make a plot of the true path of the plane

figure()

plot(states\_true(1,:),states\_true(2,:));

title('True Trajectory');

xlabel('x position in m');

ylabel('y position in m');

%% Run Filter

Cs = eye(4).\*3;

[s\_est] = NLKF1\_ver2(states\_true, init\_state, Cs,Cn, Q, B);

%% Plot the Filter Output and State

figure()

hold on

plot(states\_true(1,:),states\_true(2,:));

plot(s\_est(1,1:end-1),s\_est(2,1:end-1),'g');

hold off

%% Plot the Filter Output and State

dx = abs(states\_true(1,:))-abs(s\_est(1,:)); %x distance from tangent to true path

dy =abs(states\_true(2,:))-abs(s\_est(2,:)); %y distance from tangent to true path

figure()

hold on

plot(states\_true(1,:),states\_true(2,:));

plot(s\_est(1,1:end-1),s\_est(2,1:end-1),'g');

xlabel('x in m')

xlabel('y in m')

title('Filter Output Over Time')

legend('Modeled State','Predicted State')

hold off

figure()

subplot(2,1,1);

plot((1:N+1),dx,'k');

xlabel('Time in Sec')

ylabel('X Error in m')

title('X Error Over Time')

subplot(2,1,2)

plot((1:N+1),dy,'r');

xlabel('Time in Sec')

ylabel('Y Error in m')

title('Y Error Over Time')

# Code: Linearized Observation Model

function [linH] = lin\_H\_V2(x,y)

linH = [(x/(x^2+y^2)^.5) (y/(x^2+y^2)^.5) 0 0;(-y/(x^2+y^2)) (x/(x^2+y^2)) 0 0];

end

# Code: The Filter

function [s\_est] = NLKF1\_ver2(data\_mat, s\_init, Cs,Cn, Q, B)

%%% Simple Kalman Filter for a plane moving in two dimensions

%parameters passed:

%data\_mat has each measurement vector in a column

%Fs is the measurement frequency

%s\_init is the initial state (in this case probably zeroes)

%C is the initial covariance matrix that sets off the recursive computation

%of mean square errors. The larger the covariance, the less credence the

%filter gives to the state and the more it factors in the measurement in

%its error computations

%B: distributes unmodeled disturbances among the states

%Q: vector of mean values of unmodeled disturbances and noise

%%% In this case the state variables are X and Y position, and X and Y

%%% velocity. The measurement variables are r and theta

%initialize State transition matrix

STM = zeros(4,4);

%Predicted position in one direction = old position + Velocity(del)

%Predicted Velocity = Old Velocity

Fs = 1;

del = 1/Fs;

STM = [1 0 del 0; 0 1 0 del; 0 0 1 0; 0 0 0 1];

%Provide partial derivative of the measurement model about which to

%linearize point to point to obtain measurement matrix for each iteration of the filter. The measured variables are r and theta, or range

%initialize matrices to store various filter values

MSE\_est = Cs; %the MSE matrix

%for the Kalman gain term (to weight the difference between the prediction

%and the measurement)

Kalman\_gain = zeros(length(s\_init),length(s\_init));

s\_est = zeros(length(s\_init),size(data\_mat,2)-1);

s\_pred = zeros(length(s\_init),size(data\_mat,2)-1);

s\_est(:,1) = s\_init;

gain\_vecs = zeros(2,size(data\_mat,2)-1);

%make identity matrix for computing the estimate MSE

I = eye(length(s\_init));

% start the filter loop

for i = 1:size(data\_mat,2)-1 %%%% Note I changed this to start at 2

% make prediction

s\_pred(:,i+1) = STM\*s\_est(:,i);%state predicted by STM

Pred\_MSE = STM\*MSE\_est\*STM'+B\*Q\*B'; %Mean square error associated with the prediction

% made by the STM, which is not yet refined by how distant it is from

% the measurement

%x and y are called out to pass to the routine (lin\_H\_V3) that

%linearizes the observation model about approriate point on each

%iteration of the filter.

x = s\_pred(1,i+1);

y = s\_pred(2,i+1);

H = lin\_H\_V2(x,y);

GT = Cn+H\*Pred\_MSE\*H';

Kalman\_gain =(Pred\_MSE\*H')\*inv(GT);

gain\_vecs(:,i) = diag(Kalman\_gain);

err = H\*data\_mat(:,i+1)-H\*s\_pred(:,i);

s\_est(:,i+1) = s\_pred(:,i) + Kalman\_gain\*(err);

MSE\_est = (I - Kalman\_gain\*H)\*Pred\_MSE;

end

end