1 Introduction

Kalman filtering is a quick way of estimating the true value of a variable, by using different equations and inputs via an iterative process. In this project, we are trying to estimate the location and the speed of a vehicle which is moving in one dimension, provided the vehicle's acceleration each $100msec$ as the control variable, and GPS data as observation at roughly each three second. My implementation follows the following steps:

0) **Initialization:** The initial values for state vector,estimation error vector, and measurement error vector are given in order as:

$$X_{0|0} = \begin{bmatrix} 0m \\ 0m/s \end{bmatrix} \quad \omega_0 = \begin{bmatrix} 10m \\ 1m/s \end{bmatrix} \quad \beta_0 = \begin{bmatrix} 2.7 * 10^{-5} deg \\ 0.1m/s \end{bmatrix}$$

for covariance matrices $Q, R, \& P$ we have:

$$Q = \begin{bmatrix} 10m & 0 \\ 0 & 1m/s \end{bmatrix} \quad R = \begin{bmatrix} 2.7 * 10^{-5} deg & 0 \\ 0 & 0.1m/s \end{bmatrix} \quad P_{0|0} = Q$$

We also have: $F_t = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad B_t = \begin{bmatrix} 1 & (\Delta t^2)/2 \\ 0 & \Delta t \end{bmatrix} \quad u_t = \text{acceleration at } t$

The other steps are taken care of per iteration:

1) **Estimating the state vector:** At each iteration, using the provided acceleration, we can estimate the location and the speed of the vehicle at last $100msec$ after time being by second Newton law. This estimation comes along with an uncertainty, which we assume follows a $N(0, \sigma_p)$ distribution. The covariance matrix associated with this random variable, P will be updated each iteration at this step. This is actually to hard to derive this matrix as it deals with true state values, but for the sake of this project I assume that the errors associated with the velocity and the location are independent so their covariance iz zero.Hence, at the end of this step, P is diagonalized.

$$\hat{x}_{t|t-1} = F_t \hat{x}_{t-1|t-1} + B_t u_t \qquad (1)$$

$$P_{t|t-1} = F_t P_{t-1|t-1} F^t + Q_t \qquad (2)$$

Where Q is process noise covariance matrix[1].

2)**Calculating the Kalman gain:** Kalman gain is the heart of each Kalman filter, which determines how much the updated estimate for current state is related to the observation and how much is related to the estimated state vector. K is always between 0 and 1. The more K is, the more the updated estimated result is closed to the measurement and the less

is related to the estimated state vector at step 1.

$$K = \frac{P_{t|t-1}H_t^T}{H_t P_{t|t-1}H_t^T + R_t} \quad (3)$$

Where R is the measurement error covariance matrix which we assume follows a $N(0, \sigma_r)$ distribution and H is the transformation matrix which transforms the dimensions from the state to the measurement domain. If the units are the same then $H = I$. So, another form of interpretation of Kalman gain is:

$$K = \frac{P}{P + R} \quad (4)$$

In other words, Kalman gain measure the errors of observation and estimation and decides that the one with lower error effects the output more.

3)**Calculating current estimate:** Using the Kalman gain provided by step 2, we update our estimation of current state and the state error covariance matrix as follows:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H_t\hat{x}_{t|t-1}) \quad (5)$$

$$P_{t|t} = P_{t|t-1} - K_t H_t P_{t|t-1} \quad (6)$$

Where z_t is the observation in degrees latitude for the location and Km/H for the velocity. Equation (5) can be written in another fashion. Without any subtraction fro the generality of the problem let's assume $H = I$ to see that better.

$$\hat{x}_{t|t} = (I - K_t)\hat{x}_{t|t-1} + K_t z_t \quad (7)$$

It is interesting that the Kalman filter takes a linear combination of the measurement and the estimation to update the estimation. With this approach the derivation we made about Kalman gain in previous step is more obvious.

2 Operational Consideration

Before I get into description of the code, I need to explain three important points:

1- For the sake of easier implementation, I will transform the GPS observations into the dimension of state vectors, then will feed them to the Kalman filter and accordingly assume that $H = I$. In order to do so, I will do the following:

$$\begin{bmatrix} location(m) \\ velocity(m/s) \end{bmatrix} = \begin{bmatrix} R_{lat} + HEIGHT & 0 \\ 0 & \frac{1}{3.6} \end{bmatrix} \begin{bmatrix} latitudee_1(deg) - latitude_{e_1}(deg) \\ velocity(Km/H) \end{bmatrix}$$

Where $HEIGHT = 54m$ and $latitude_{e_0} = 38.094226^{\text{deg}}$

2- As the observations come along at random times at most of the iterations the Kalman filter does not have any measurement values. In this cases the filter has to decide just based on the result of step 1. In other words, the estimation of the current state will not be updated or equivalently based on (7) $K_t = 0$. In order to make K_t we can push R to infinity and this leads to $P_{t|t} = P_{t|t-1}$. So, practically, in these iterations we will pass out the estimated state vector as the output of Kalman filter and will not update the error covariance matrix created by step 1 as well.

3) Both the estimation and the measurement have uncertainties associated to them but, as the value of uncertainty is not deterministic, practically we will take the expectation of that which is zero, so no error variable will be introduced to our simulation.

3 Code Blocks

In order to run the code and see the results, please replace your files with the similar files in the directory. Then simply run the code form the **newMain** block. The codes are commented clearly and in this section we will show how it works. First of all, the main block calls **extractData** block and creates 4 double arrays including acceleration, GPS time, GPS observed location, and GPS observed velocity.

The original values are fed into the program using **feedOriginal** module.Then the measurement error covariance matrix will be transformed to the dimensions of the state vector, using **calcR** function. GPS-time will be altered to form **fixedTime** array each element of which is the index of one of the iterations in which a measurement is introduced; for example the observation which comes along 29.003 seconds after the start of simulation will correspond to number 29 of fixedTime array and will be used to update the estimation for the state of 30th step. As mentioned above $H = I$. There are two arrays named as **holdP** and **holdK** which keep track of matrix P and Kalman gain to analyze the performance of Kalman filter. Then for all 3001 acceleration values, the **predictNext** function will estimate the state vector and updates the state error covariance matrix. When **fixedTime** array shows that there is a measurement value available, the **calcGain** module will calculate the Kalman gain, the **calcZ** function will transform the observation to the domain of state vector, and **updateNext** function will update the estimation and updates the estimation error covariance matrix outputted from step 1 as well. At the end the results are plotted in the format mentioned in the description of the project. The other modules serve the main module and just code the mathematical formulations described in previous sections.

Please note that as we go on the Kalman gain diminishes, which means that the estimations are getting more and more accurate and the filter will take less and less effect of the observations. Also, the variance of error has a diminishing behavior which again means the level of uncertainty is decreasing.

4 Graphs

Without making any changes in the values of R and Q we will have the following graphs: It can be seen that without presence of measurement the estimation starts drifting away but when measurement exists, it brings the estimation back on track.

Figure 1

The final P and K are:

$$P = \begin{bmatrix} 297.0682 & 0 \\ 0 & 29.0996 \end{bmatrix} \quad K = \begin{bmatrix} 0.9896 & 0 \\ 0 & 0.9965 \end{bmatrix}$$

Which show that this filter is giving more weight to the measurements and errors do not converge.

In the above graph, since the error covariance of process noise is way bigger than the error covariance of measurement noise, the kalman filter puts more weight on the data from measurement. In order to make our estimation less uncertain, we need to manipulate R and Q to adjust the level of uncertainty in process and measurement noises. In fact, there are multiple advanced ways of tuning a kalman filter. Here, I tried to change the sensitivity of the filter to the measurements and estimations by trial and error, given the fact that, after long runs of measurement absence, the uncertainty of the estimations will grow iteration to iteration; but in practice Q is so small with respect to P, otherwise it means that the Kalman filter is not predicting good. So, if we change Q to a smaller value, then both K and P converge, and our estimations get better and better. I changed Q to be P/1000 in each iteration in **predictNext** module. Here is the corresponding graph:

Figure 2

The final P and K are:

$$P = \begin{bmatrix} 0.1202 & 0 \\ 0 & 0.0032 \end{bmatrix} \quad K = \begin{bmatrix} 0.0385 & 0 \\ 0 & 0.0312 \end{bmatrix}$$

Which show that the filter's estimations are improving and being more valuable as times go on.

5 References

[1] Ramsey Faragher, "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation," IEEE Signal Processing Magazine, vol. 29. no. 5, pp. 128-132, September 2012.