KALMAN FILTER

Kalman filtering is an algorithm that provides estimates of some unknown variables given the measurements observed over time. Kalman filters have been demonstrating its usefulness in various applications. It has relatively simple form and require small computational power. Most of the modern systems are equipped with numerous sensors that provide estimation of hidden (unknown) variables based on the series of measurements. For example, the GPS receiver provides the location and velocity estimation, where location and velocity are the hidden variables and differential time of satellite's signals arrival are the measurements. It is essentially an estimation algorithm, which takes into account the current measurement and uncertainties in both, the estimate and measurement to assign appropriate weight to both while making its prediction.

It has been used in a tremendous amount of situations which range from navigation to signal processing and econometrics. It is actively being used in the field pf robotics such as motion planning.

Kalman filter is used to estimate the state variables of linear dynamic systems. By Linear we mean that the state variables can be described using linear equations involving each other. If non-linearity is involved, the basic form of Kalman filter does not make very good prediction. It has to be extended to handle non linearity. It works in a two step process The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error,

including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. The algorithm is recursive. It can run in real time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required.

One of the main assumptions of Kalman filter is that it assumes that the errors are random and follow a gaussian distribution. Though regardless of Gaussianity, if the process and measurement covariances are known, the Kalman filter is the best possible linear estimator in the minimum mean-square-error sense.

TECHNICAL DESCRIPTION

Kalman filters are based on linear dynamical systems discretized in the time domain. The state of the target system refers to the ground truth (yet hidden) system configuration of interest, which is represented as a vector of real numbers. At each discrete time increment, a linear operator is applied to the state to generate the new state, with some noise mixed in, and optionally some information from the controls on the system if they are known. Then, another linear operator mixed with more noise generates the measurable outputs (i.e., observation) from the true ("hidden") state.

In order to apply Kalman filter to a particular scenario, we must capture the dynamic model of our application in the form that Kalman filter requires. To be more specific, we must model the equations governing our application in the form of matrices.

The different components are described below

$\mathbf{F}_k$, the state-transition model. This matrix helps us to predict the value of current state variables using the previous state variables.

$\mathbf{G_k}$ the control matrix. This matrix helps us to incorporate the changes in the current state variables due to the control inputs that we might receive as we move ahead with the time.

$\mathbf{W_n}$ – The process noise vector. Contains the error we might have due to the modelling process. The model might not be able to exactly predict the real state variables.

$\mathbf{P_{n,n}}$ Estimates the current uncertainty in the particular state. This is a covariance matrix and is captures the estimate uncertainties.

Q Captures the process noise covariance. This is also a covariance matrix which captures the covariance of noise in the process estimation.

R Captures the covariance in the measurement. This covariance matrix captures the errors and uncertainties in the measurement process.

$\mathbf{K_n}$ This represents the Kalman gain. This decides how much weight we should give to our estimate and measurement based on their uncertainties.

These matrices are configured specific to a particular situation. Some of them might change as we get more measurements. For example, as we progress the uncertainties in the estimation keep on decreasing.

Kalman Filter works in 2 steps. The "***predict***" and the "***update***" step. There are a total of 5 equations in the Kalman filter.

STATE ESTIMATE EQUATION

$$Xn = F * Xn + G * U_n$$

X_n is the state estimate of the previous iteration and U_n are the control inputs at the current instant. This equation captures the dynamics of the model and the F and G matrices are problem specific. For example, if we consider the case of applying the Kalman filter to an object falling under gravity and we are trying to predict the position and velocity of the object.

X_n is a 2X1 vector $\begin{bmatrix} x \\ V \end{bmatrix}$

F is a matrix of size 2X2 $\begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$

G is a 2X1 matrix $\begin{bmatrix} 0.5\Delta t^2 \\ \Delta t \end{bmatrix}$

These matrices model the dynamics of the system. Substituting these into the equation will give us the equations that govern the particular system that we are applying Kalman filter to.

COVARIANCE EXTRAPOLATION

$$P_{n+1,n} = F*P_{n,n}*F^T + Q$$

This is the prediction of covariance matrix for the next instance given the covariance matrix of the particular instance.

These two equation form the "**PREDICTION**" step of the particular instance. The remaining 3 equations form the "**UPADATE"** step of that particular instance.

STATE UPDATE EQUATION

$$X_{n,n} = X_{n,n-1} + K(Z_n - HX_{n,n-1})$$

This equation estimates the state variables for current state using the previous information assigning appropriate weight to the each, estimation and measurement, based on their uncertainties. Z_n is the measurement that this particular step took. The Matrix H is just a conversion matrix showing that we might not be measuring all the state variables that we are trying predict.

KALMAN GAIN

$$K_N = P_{n,n-1}H^T(HP_{n,n-1}H^T + R)^T$$

This helps us decide the weight assignment. If the K_n is high, it means that the estimation uncertainty is high, hence more weight is given to the measurement and vice versa.

COVARIANCE UPDATE EQUATION

$$P_{n,n} = (I-K_nH) *P_{n,n-1} *(I-K_nH)^T + K_nRK_n^T$$

This equation estimates the current estimate covariance which is further used to predict the covariance for the next step. A simpler for of the equation is also used, but this form of the equation is more stable as with respective non-optimal Kalman gain as compared to other form. In other word, small floating point computational errors do not affect this form much as compared to the other form. The simpler form of the equation is

$$\mathbf{P_{n,n}=(I-K_nH)\ *P_{n,n-1}}$$

We can see that the values in $P_{n,n}$ keeps on decreasing at each step proving that we become more and more certain about our prediction as we predict more. To make this clearer consider the case of single dimension. The $P_{n,n}$ will be a single value so will be the Kalman gain. Therefore $P_{n,n}= (1-K_n)*P_{n,n-1}$ hence we can see that the uncertainty keeps on decreasing continuously.

WORKING PROCEDURE

The process starts with initializing the state vector and the estimate covariance matrix. All the remaining matrices except the Kalman gain do not change. After initializing, we make a prediction , i.e we carry out the prediction step. After the prediction step, we start our iterations. At each iteration, we first carry out the update step, in which we calculate the Kalman gain , the current state variables and the covariance matrix and then carry out the prediction step.

IMPLEMENTATION DETAIL

Our implementation of the Kalman filter works requires the user to enter the required matrixes as input in the form of files. Hence it is generic and can work on any problem type. We have used an external library called as "**EIGEN**" which is extremely efficient when it comes to matrix operations. Hence the user need to have the files for this library and has to provide path to the folder of this library while executing.

While executing, we take in 2 arguments on the command line, the name of 2 files. The first argument is the file that contains the matrices and the second argument is the file that contains all the data for each step, i.e. the measurements and the control inputs. The format of the matrix file is as follows.

In the beginning, the file contains 3 numbers n, u, z.

n- the number of entries in the state vector

v- number of elements in the control input vector

z- the number of elements in the measurement vector.

The rest of the files contains the matrices in the following order.

F,G,H,P,Q,R

The sizes of these matrices are depended on the 3 values at the beginning of the file and can e determined as soon as we read the 3 values. Therefore, we do not need to specify the sizes of the matrices.

The second file contains the data for the Kalman filter. The first entry is the initial value of X (the state vector) and the next entry is the initial

value of U, the control input vector . after these, there is a number which denotes the number of measurements. After this number there is the data in the following format.

Z_i U_i

First is the measurement vector at a particular instant and followed by the control input vector at that instant.

CODE

We have created a class called KF which stores all the required matrices and vectors as its data members. The datatype of these matrices and vectors is "**MatrixXd** " which is the Matrix datatype of the library Eigen.

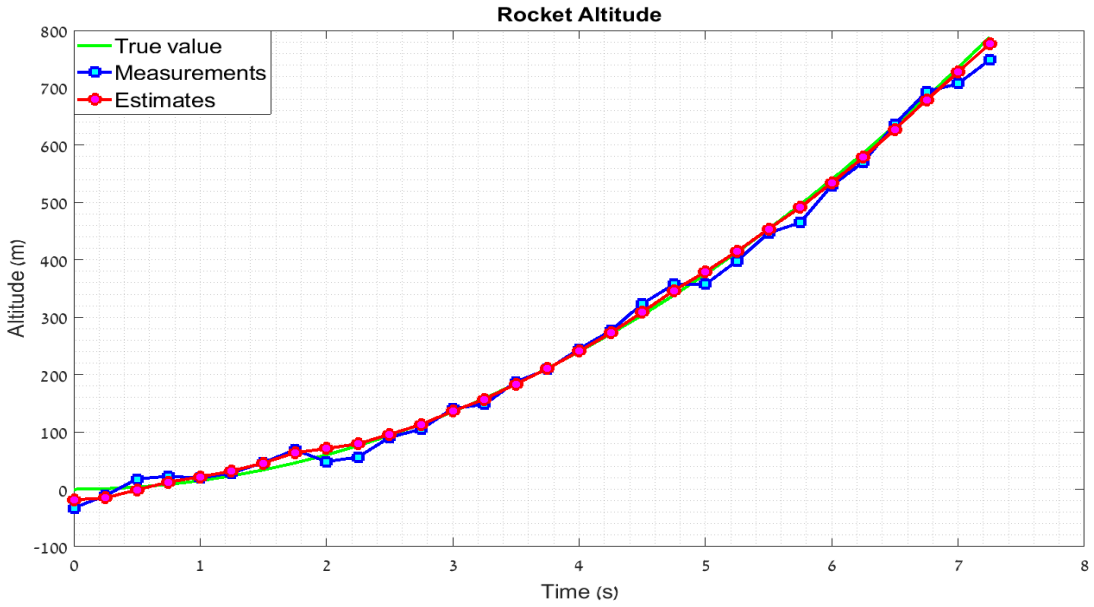
We first read the matrices from the matrix file by calling the function "**read_mats**", the file was given as the first argument at the command line during execution. Then we initialize the 'X' and 'U' vector by calling the function "**init_x_U**" the values are read from the data file which was the second command line argument. The same file is than used to read data by calling the "**read_DATA**" function.

The State vector, the measurement vector and the control input vector are the class variables and are updated for each iteration. All the required are also the class variables. The computation of Kalman filter are divide into 2 functions "**predict**" and "**update**" . The predict function implements the 2 equations for the predict part. And the update function implements the 3 equations for the update. All the variable in the equations are matrices, the library Eigen takes care that the multiplication operator results in a matrix multiplication, hence the equations look like normal equations, even though the variables are matrices.

The “**solve**” function calls the predict and update function repeatedly depending on number of data entries.

RESULTS

We ran the code on an test case of estimating the altitude of a rocket, given that it is accelerating at a uniform acceleration upwards



This image was taken from here. They applied the Kalman filter on the same example. Our results are similar to them hence the graph comes out to be same.

APPLICATIONS OF KALMAN FILTER

As mentioned before, Kalman filter is extensively used in navigation and some other domains such as robotics as well.

1) GPS. Kalman filter is extremely useful in determining the position and velocity of the vehicle using the measurements from the sensors like, accelerometer, odometer and the readings from the gps. Kalman filter can be applied to these and the position and the velocity and be effectively estimated keeping in mind the measurements and the uncertainties of different sensors.

2) RADARS. Kalman filters also find their use in RADARS. RADARS are used to estimate the position of an object based on the time taken by a light beam to bounce back from the object. These measurements can be noisy and hence Kalman filter is an effective way to produce estimates to the positions using the readings such as the time taken by the beam to return.

3) ECONOMETRICS. Kalman filters have also been used in the field of finance and econometrics. Financial data have been observed to have certain regularities in statistical properties. It is essentially a least squares (Gauss Markov) procedure and therefore gives Minimum Mean Square Estimators, with the normality assumption. Not only is it used directly in economic problems that can be represented in state-space forms, it is used in the background as part of several other estimation techniques, like the Quasi-Maximum Likelihood estimation procedure