The Kalman Filter

AN INTRODUCTION WITH INTUITIVE EXAMPLES

P G MAHADEVAN

SHUBHAM SAKET

EXECUTIVE SUMMARY

Developed by Rudolf E. Kalman in 1960, he describes a recursive solution to the discrete data linear filtering problem in his paper. Also known as Linear Quadratic Estimation (LQE), this has been subject to extensive research and application, particularly in the area of autonomous or assisted navigation since it was published.

The purpose of this document is to provide a basic understanding of the Kalman Filter, with simple practical examples for laying the foundation bricks of this vast topic.

SYNOPSIS

Basically, Kalman Filter is a linear, discrete time, finite dimensional time-varying system that evaluates the state estimate that minimizes the mean-square error. The filter dynamics result from the consecutive cycles of prediction and filtering. In simple terms, Kalman filtering is an iterative mathematical process that uses a set of equations and consecutive data inputs to estimate the values we are interested in associated with the object *quickly*.

A physical system, (e.g., a mobile robot, a chemical process, a satellite) is driven by a set of external inputs or controls and its outputs are evaluated by measuring devices or sensors, such that the knowledge on the system's behaviour is solely given by the inputs and the observed outputs. The observations convey the errors and uncertainties in the process, namely the sensor noise and the system errors. Based on this available information, it is required to obtain an estimate of the system's state that optimizes a given criteria. This is the role played by not only Kalman filter, but also a general 'filter'.

The following terms will be used every now and then henceforth:

- **Observed/Measured value:** It is simply the value of the measurand given by a measuring instrument.
- **Error/Uncertainty in a value:** It is the doubt that exists about the value. It has the following two aspects:
    - The width of the margin, or 'interval'. This is the range of values one expects the true value to lie within, not necessarily the range of values one might obtain, the measured value may include outliers too.
    - Confidence level, i.e. how sure the experimenter is that the true value lies within the margin.
- **True value:** The actual value of the measurand without the error factor. It is defined as the value obtained after an infinite series of measurements performed under the same conditions with an instrument not affected by systematic errors.

A more practical and pragmatic definition may go like: a value compatible with the definition of a given particular quantity.

- **Predicted value:** The value that a variable is supposed to take at a particular instant under known conditions. We may formulate equations to predict a variable under given conditions. This value is also prone to certain errors, like we might have missed a condition that affects the value being predicted while formulating the equations.

In the filtering process during one iteration, the filter typically has the observed data at any particular time interval, the predicted value at that particular instant, and the errors in observation and prediction as inputs. With this input data, the Kalman gain is calculated, which is a weighted measure to estimate the true value. We know that in real life scenario, neither the observed value is perfect (it might contain some noise), nor the predicted value is accurate (no data can be fitted to a perfect curve, some residual is always there). So, to get the nearest estimate to the true value, we need to decide how much emphasis we need to give to the observed value, and how much to the predicted value, since predicted value is theoretically accurate and observed value is the practically measured value. Kalman gain does this task. As said earlier, it is a measure of how much the observed and the predicted values contribute to the estimation of true value.

Kalman gain takes care of the following:

- High deviation of measured/predicted values from the true value does not throw us “off the track”, being fooled by erroneous measurements/predictions.
- Estimating the true value pretty accurately in the initial iterations itself.

Note: To make the filtering process simpler and for avoiding the effort of prediction of values for simpler problems where we can afford marginal residuals, we feedback the filter with the previous estimated value in place of the predicted value and calculate the current estimate.

THE PROCESS

The Kalman filtering process (passing previous estimate as feedback) involves the following three major steps:

- Calculation of Kalman gain (K) using the error in estimate (E_{EST}) and the error in measurement (E_{MEA}).

$$K = \frac{E_{EST_{(t)}}}{E_{EST_{(t)}} + E_{MEA}}$$

(Note that t denotes the current time instant.)

Intuitively, by looking at the formula above, we can say that *higher the Kalman gain, more is the weight placed on the estimate than the measurement.*

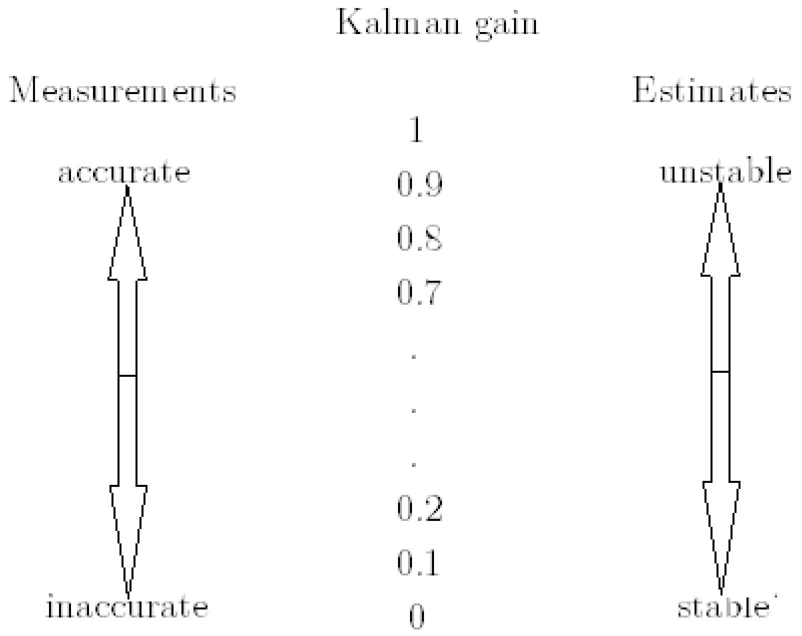

Fig 1: Kalman gain: its intuitive value

- Update the previous estimated value ($EST_{(t-1)}$) with the new value ($EST_{(t)}$), giving appropriate weightage to the measured value (MEA) and the previously estimated value using Kalman gain.

$$EST_{(t)} = EST_{(t-1)} + K[MEA - EST_{(t-1)}]$$

- Now that we have updated the estimated value, error in the estimated value will also change as follows:

$$E_{EST_{(t)}} = [1 - K](E_{EST_{(t-1)}})$$

Since we know the process now, let us solve a simple example to illustrate the steps involved.

CASE STUDY – 1: TEMPERATURE ESTIMATE

We'll apply Kalman filtering to a simple univariate data to estimate the value, and let us see whether we are able to approximate the true value.

Consider the following temperature data of a city, measured in Fahrenheit, in constant time intervals on the same day (measurement conditions are assumed to be same).

Measured values (MEA): 75, 71, 70, 74

Error in measurement (E_{MEA}): ±4

True value: 72

For starting with the filtering process, we need an initial estimate and error in estimate. Since the readings start from time instance (t), we do not have an estimate for the instance (t-1). One advantage of Kalman filtering is, we can assume any value for the initial estimate, and Kalman gain will adjust the value in such a way, that the estimated value will start getting closer to the true value in the initial iterations itself. This is what a Kalman filter is meant to do.

Let us assume the following values:

Initial estimate (EST): 68

Estimation error (E_{EST}): 2

Step 1: Calculation of Kalman gain.

$$K = \frac{E_{EST_{(t)}}}{E_{EST_{(t)}} + E_{MEA}}$$

$$K = 2/(2+4) = 0.33$$

Step 2: Update the estimated value.

$$EST_{(t)} = EST_{(t-1)} + K[MEA - EST_{(t-1)}]$$

$$EST_{(t)} = 68+0.33[75-68] = 70.31$$

Step 3: Update the error in estimate.

$$E_{EST_{(t)}} = [1-K](E_{EST_{(t-1)}})$$

$$E_{EST(t)} = [1-0.33](2) = 1.34$$

If we tabulate the results after the first iteration:

Iteration #	Time	MEA	E_{MEA}	EST	$E_{EST(t-1)}$	K	$E_{EST(t)}$
0	t-1			68	2		
1	t	75	4	70.31		0.33	1.34

After the first iteration itself, we can see that the estimated value is approaching the true value (72). Even though the measured value (75) was quite high, in fact higher than the true value, the estimated value was adjusted to 70.31 after giving appropriate weightage to the measured value and the previous estimated value.

In the next iteration, we use the value estimated in iteration 1, and proceed as follows:

$$K = 1.34/(1.34+4) = 0.25$$

$$EST_{(t)} = 70.31+0.25[71-70.31] = 70.48$$

$$E_{EST(t)} = [1-0.25](1.31) = 0.98$$

We proceed similarly, and calculate the estimated value up to time (t+3), since we have only 4 measured values. Results have been tabulated below:

Iteration #	Time	MEA	E_{MEA}	EST	$E_{EST(t-1)}$	K	$E_{EST(t)}$
0	t-1			68	2		
1	t	75	4	70.31		0.33	1.34
2	t+1	71	4	70.48		0.25	0.98
3	t+2	70	4	70.38		0.20	0.78
4	t+3	74	4	70.96		0.16	0.66

From the above results, see how we started from 68, after the first iteration, the value was estimated as 70.31, with an error of ±1.34. In the next set of iterations, the estimated value did not vary hugely with the measured value. As we proceeded with the iterations, the error in estimated value decreased, which shows that the filter was becoming more confident with the estimated value. After 4 iterations, we got the estimated value as 70.96, with an error of ±0.66, which is quite close to the true value 72. We can get more precise results with more iterations.

Now that we have seen a basic example and have an idea of the process, we are ready to plunge into the depths, and solve a real time problem.

KALMAN FILTER – THE MULTI-DIMENSION MODEL

As some of us might be doubting, Kalman filter is not as sweet and simple as illustrated in the example above, nevertheless, the principle remains the same. We need to modify the above formulae in such a way, that they can be applied to real world scenarios, where we use Kalman filters to predict the motion of a falling object to the motion of a rocket bound towards the orbit of a planet. Another real time example might be the prediction of the motion of an airplane. We might be getting constant readings about the position and velocity of the plane from the radars in periodic intervals, and we might want to keep updating where the plane might be sometime later, with the help of predictions and actual measurements, with the help of Kalman gain.

In such scenarios, the external forces acting on the object play a vital role, gravity in this case, acts in a supportive way for a falling object and influences its motion, and in case of a rocket launch, we need to overcome the gravitational force acting on the rocket (weighing tonnes) in order to launch it in a direction opposite to the gravitational pull. Physics comes into action here, and influences the object's motion greatly - Remember *escape velocity?* In a sense, the dependent variables are plenty in real time scenarios, and we also need to consider the noise in measurements. Talking about mathematical calculations, representing the equations in matrix form comes handy, which you'll see why in the forthcoming example.

The following diagram will give an overview of the real – life Kalman filter.

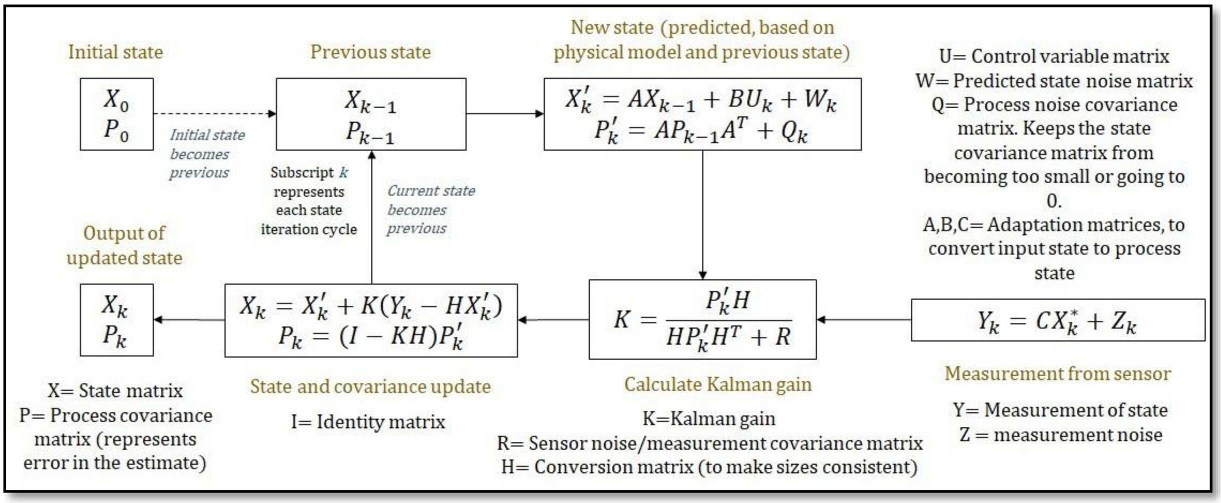
Initial state
X_0 P_0
Initial state becomes previous
Previous state
X_{k-1} P_{k-1}
Subscript k represents each state iteration cycle
Current state becomes previous
New state (predicted, based on physical model and previous state)
$X'_k = AX_{k-1} + BU_k + W_k$
$P'_k = AP_{k-1}A^T + Q_k$
U= Control variable matrix
W= Predicted state noise matrix
Q= Process noise covariance matrix. Keeps the state covariance matrix from becoming too small or going to 0.
A,B,C= Adaptation matrices, to convert input state to process state
Output of updated state
X_k P_k
X= State matrix
P= Process covariance matrix (represents error in the estimate)
$X_k = X'_k + K(Y_k - HX'_k)$
$P_k = (I - KH)P'_k$
State and covariance update
I= Identity matrix
$K = \frac{P'_k H}{HP'_k H^T + R}$
Calculate Kalman gain
K=Kalman gain
R= Sensor noise/measurement covariance matrix
H= Conversion matrix (to make sizes consistent)
$Y_k = CX^*_k + Z_k$
Measurement from sensor
Y= Measurement of state
Z = measurement noise

Fig 2: Kalman filter: an overview

In the initial state, we have the state matrix and the Process covariance matrix.

State matrix (X): The initial and the updated state matrices consist of the values of interest, which undergo changes on the basis of measured values and the Kalman gain, and the new value is predicted, which follows the same cycle again. These values of interest may be the position and velocity of an object in one, two or all the three dimensions.

Process covariance matrix (P): Consists of the error due to the process.

With these two matrices, we theoretically predict the new state of the object. For predicting the new state, we take the Control variable matrix

Control variable matrix (U): A matrix containing the variables that control or influence the value of interest.

Adaptation matrices (A, B, C): Matrices used to convert from one form to the other.

Noise matrices (W, Q, Z, R): W is the Predicted state noise matrix and Q is the Process noise covariance matrix. Basically, both contribute to the noise/errors in predictions. Z is the noise/uncertainty in measurement, and R is the sensor noise covariance matrix, which constitutes the measurement error.

Measured values (Y): The Y matrix constitutes the measured values of the state.

Kalman gain (K): With the predicted value and the measured value, we come up with the Kalman gain, which decides how much of the estimate we need to impart on the measurement, and how much on the predicted value.

Conversion matrix (H): Used to bring the matrix to the desired form without changing its values.

The following example of a falling stone will illustrate the calculation. We will consider the motion of the stone only in one dimension (along the y - axis) for ease of calculation. The calculation process will remain the same even for variables in more than one dimension, the only difference being that we'll be having more values in the matrix. Instead of dealing with a 2x1 matrix, we'd be dealing with a 4x1 matrix or even higher.

CASE STUDY – 2: MOTION OF A FALLING STONE

Initial position $y_0 = 20$m, initial velocity $\dot{y}_0 = 0$m/s, at instant t=1s, y_1=16m, $\dot{y}_1$=-10m/s

Observation errors: $\Delta y = 0.6$m, $\Delta \dot{y} = 0.03$m/s

We start with the state matrix for a falling object, along the y-axis. The state matrix will typically look like:

$$X = \begin{bmatrix} y \\ \dot{y} \end{bmatrix}$$

where y is the position and $\dot{y}$ is the velocity of the object being tracked at that particular state.

In 2-dimensions, the matrix would look like:

$$X = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}$$

and similarly in 3-dimensions, we'll have the position and velocity along all the 3-axes, but since we are considering the motion in 1-dimension, we'll stick with the previous state matrix having position and velocity along y-axis only.

Step-1: Calculate the predicted state.

We'll start with the following equation:

$$X_k = AX_{k-1} + BU_k + W_k$$

where we update the state matrix based on the previous state and the control variable matrix. For updating the state matrix, we borrow the following two equations from the Newton's laws of Kinematics:

$$x = x_0 + \dot{x}_i t + \frac{1}{2}\ddot{x}t^2$$
$$\dot{x}_f = \dot{x}_i + \ddot{x}t$$

In accordance with these kinematics equations, we formulate the adaptation matrices A and B, and the control variable matrix u, since we know that for our case, the control variable matrix would consist of acceleration due to gravity (g), which brings about a change in the position and velocity of the object.

$$A = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix}, u = \begin{bmatrix} g \end{bmatrix}$$

Substituting these values in the state matrix update equation (considering the value of noise to be 0), we get the equation for calculating the updated state as follows:

$$X_k = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta T^2 \\ \Delta T \end{bmatrix} \begin{bmatrix} g \end{bmatrix} + 0$$

On working out further, we get the following:

$$X_k = \begin{bmatrix} y_{k-1} + \Delta T.\dot{y}_{k-1} \\ 0 + \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}\Delta T^2.g \\ \Delta T.g \end{bmatrix}$$

$$X_k = \begin{bmatrix} y_{k-1} + \Delta T.\dot{y}_{k-1} + \frac{1}{2}\Delta T^2.g \\ \dot{y}_{k-1} + \Delta T.g \end{bmatrix}$$

Substituting the values in the above equation as per the question statement ($y_{k-1} = 20$, $\dot{y}_{k-1} = 0$) for $\Delta T = 1$, $g = -9.8$ m/sec^2:

$$X_k = \begin{bmatrix} 20 + (1)(0) + \frac{1}{2}(1)^2(-9.8) \\ 0 + (1)(-9.8) \end{bmatrix} = \begin{bmatrix} 15.1 \\ -9.8 \end{bmatrix}$$

Hence we get the predicted value of the state 1 second later. Similarly, we can calculate the position and velocity for the next time periods, and verify with the help of the Kinematic equations.

Step-2: Come up with the initial process covariance matrix.

Now, we also need the process covariance matrix, which constitutes the variance and the covariance of the values being measured. This typically is due to the deviation from ideal values because of the process, and not because of any measurement errors, which would be considered as noise. For the falling stone, the measured values may not be exactly equal to the predicted values, different other forces like the thrust of wind might be acting on the stone, decelerating/promoting its motion. We consider the variance and covariance of the values, since these give an estimate of how much a value can vary from its mean, which would further be used in the calculation of Kalman gain to incorporate the effect of external forces and update the values.

For our case, the covariance matrix P is given by:

$$P = \begin{bmatrix} \sigma_y^2 & \sigma_{y\dot{y}} \\ \sigma_{\dot{y}y} & \sigma_{\dot{y}}^2 \end{bmatrix}$$

Covariance and variance are given by:

$$Variance\ \sigma_y^2 = \frac{\sum_{i=1}^{N} (\overline{y} - y_i)^2}{N}$$

$$Covariance\ \sigma_{y\dot{y}} = \frac{\sum_{i=1}^{N} (\overline{y} - y_i)(\overline{\dot{y}} - \dot{y}_i)}{N}$$

where $\overline{y}$ and $\overline{\dot{y}}$ are the mean/averages of y and $\overline{y}$ respectively, and N is the number of observations.

Not going into details of calculation of variance and covariance, for our example, let's have the standard deviations of y and $\dot{y}$ as 0.5m and 0.2m/s respectively. Since variance is standard deviation squared, we'll have the variances as 0.25 and 0.04. If the estimated error for one variable y (position) is completely independent of the other variable $\dot{y}$ (velocity), the covariance elements in the covariance matrix are taken as 0. This means that no adjustments are made to the estimates of one variable due to the process error in the other variable. So for our case, the process covariance matrix becomes:

$$P_{k-1} = \begin{bmatrix} 0.25 & 0 \\ 0 & 0.04 \end{bmatrix}$$

Step-3: Calculate the predicted process covariance matrix.

The predicted process covariance matrix is given by:

$$P_{kp} = AP_{k-1}A^T + Q_k$$

considering noise Q_k to be 0, we have all other values. Let's straight away plug-in the values and get the result.

$$P_{kp} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.25 & 0 \\ 0 & 0.04 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \Delta T & 1 \end{bmatrix} + 0$$

For the first time period (ΔT = 1),

$$P_{kp} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0.25 & 0 \\ 0 & 0.04 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + 0$$
$$P_{kp} = \begin{bmatrix} 0.25 & 0.04 \\ 0 & 0.04 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} + 0$$
$$P_{kp} = \begin{bmatrix} 0.29 & 0.04 \\ 0.04 & 0.04 \end{bmatrix}$$

Ignoring the covariance as before, we get

$$P_{kp} = \begin{bmatrix} 0.29 & 0 \\ 0 & 0.04 \end{bmatrix}$$

Step-4: Calculate the Kalman gain.

Kalman gain is given by the equation:

$$K = \frac{P_{kp}H^T}{HP_{kp}H^T + R}$$

Since we are dealing with 2x2 matrices, the predicted process covariance matrix is also 2x2, the calculated Kalman gain will also be of the same form. Since there is no form conversion required, we'll take H as an identity matrix. R is the matrix of squared errors in the observations. As per the question statement, we have the errors Δy = 0.6m, $\Delta \dot{y}$ = 0.03m/s. Forming the error/noise matrix,

$$R = \begin{bmatrix} 0.36 & 0 \\ 0 & 0.09 \end{bmatrix}$$

Substituting the values and evaluating,

$$K = \frac{\begin{bmatrix}0.29 & 0\\ 0 & 0.04\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}0.29 & 0\\ 0 & 0.04\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}+\begin{bmatrix}0.36 & 0\\ 0 & 0.09\end{bmatrix}}$$

$$K = \frac{\begin{bmatrix}0.29 & 0\\ 0 & 0.04\end{bmatrix}}{\begin{bmatrix}0.65 & 0\\ 0 & 0.13\end{bmatrix}}$$

$$K = \begin{bmatrix}0.45 & 0\\ 0 & 0.31\end{bmatrix}$$

Step-5: Bringing in the new observation.

Now, we need to bring into picture the actual measurements. We use the following equation and bring the measurements to the required form:

$$Y_k = CY_{k_m} + Z_k$$

Y_k in the above equation represents the observed state of the stone. It has to have the same format as that of the state matrix. C matrix allows us to transform the state into how we want it to be. Since the state matrix was a 2x1 matrix with the position and velocity of the stone along y axis, and we are measuring both the values, we can put both the measured values in the same form, and consider C as an identity matrix. Note that Z_k here is the noise in measurement, and this is different from R matrix. Z_k typically consists of the errors induced due to noise/fault in the measuring instrument. For our convenience here, we assume this value to be 0.

From the question statement, we have the measured position and velocity at t=1. We find the Y_k matrix as follows:

$$Y_k = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}16\\ -10\end{bmatrix} + 0$$

$$Y_k = \begin{bmatrix}16\\ -10\end{bmatrix}$$

Step-6: Calculate the current state adjusting the measured and the predicted values based on Kalman gain.

At this step, we have the predicted value, the measured value and the Klaman gain for the falling stone. Remember that if the Kalman gain is large, we have small assumed error in the measurement, and we want to put more importance on the measured value. Similarly when the Kalman gain is small, we emphasize more on the predicted values. We have the following equation:

$$X_k = X_{k_p} + K[Y_k - HX_{k_p}]$$

H is an identity matrix here, since no matrix form conversion is required. Substituting and evaluating,

$$\begin{aligned}
X_k &= \begin{bmatrix} 15.1 \\ -9.8 \end{bmatrix} + \begin{bmatrix} 0.45 & 0 \\ 0 & 0.31 \end{bmatrix} \left(\begin{bmatrix} 16 \\ -10 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 15.1 \\ -9.8 \end{bmatrix} \right) \\
X_k &= \begin{bmatrix} 15.1 \\ -9.8 \end{bmatrix} + \begin{bmatrix} 0.45 & 0 \\ 0 & 0.31 \end{bmatrix} \begin{bmatrix} 0.9 \\ -0.2 \end{bmatrix} \\
X_k &= \begin{bmatrix} 15.1 \\ -9.8 \end{bmatrix} + \begin{bmatrix} 0.41 \\ -0.06 \end{bmatrix} \\
X_k &= \begin{bmatrix} 15.51 \\ -9.86 \end{bmatrix}
\end{aligned}$$

See how the values got adjusted? This is the current state of the falling stone at t=1 as estimated by the Kalman filtering process. Since this is an iterative process, the current estimate will become the input for the next iteration, with the updated process covariance matrix, which will be calculated in the next step.

Step-7: Update the process covariance matrix.

Since the state value has been updated with the Kalman gain, the errors in the process, and hence the process covariance matrix will also get updated with the following equation:

$$P_k = (I - KH)P_{k_p}$$

Substituting and evaluating,

$$\begin{aligned}
P_k &= \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.45 & 0 \\ 0 & 0.31 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right) \begin{bmatrix} 0.29 & 0.04 \\ 0.04 & 0.04 \end{bmatrix} \\
P_k &= \begin{bmatrix} 0.55 & 0 \\ 0 & 0.69 \end{bmatrix} \begin{bmatrix} 0.29 & 0 \\ 0 & 0.04 \end{bmatrix} \\
P_k &= \begin{bmatrix} 0.16 & 0 \\ 0 & 0.03 \end{bmatrix}
\end{aligned}$$

Notice how the values got decreased in the process covariance matrix. This commonly means that the Kalman filter has narrowed down the estimate, and it is quite sure about its prediction.

Thus, the iterations continue following the 7 steps presented above, until the last measured value we have. Confidence of the Kalman filter increases and hence the estimated process error decreases on each iteration. The value of the states estimated by the Kalman filter would not vary much even if the measured value lies quite out of bounds, this is what Kalman filter is meant for.

APPLICATIONS

Kalman filters, though being an old technique, is still being used in wide range of modern applications, few to state:

- Autopilot
- Battery State of Charge (SoC) estimation
- Nuclear physics: single photon emission computed tomography image restoration
- Visual odometry
- Weather forecasting
- Economics, in particular macroeconomics, time series analysis, and econometrics
- Seismology
- 3D modelling, and much more

YOU MIGHT ALSO BE INTERESTED IN

- Ensemble Kalman filter
- Fast Kalman filter
- Invariant extended Kalman filter

YOU MAY ALSO READ

- Grewal, M. S., & Andrews, A. P. (2008). *Kalman Filtering: Theory and Practice Using MATLAB*. Wiley-IEEE Press.
- Welch, G., & Bishop, G. (2006). An Introduction to the Kalman Filter, 1–16.
- Lacey, T. (n.d.). Tutorial : The Kalman Filter, 133–140.
- Ribeiro, M. I. (2004). Kalman and Extended Kalman Filters : Concept , Derivation and Properties, (February).
- Sinopoli, B., Member, S., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M. I., … Sastry, S. S. (2004). Kalman Filtering With Intermittent Observations, *49*(9), 1453–1464.
- Mcgee, L. A., & Schmidt, S. F. Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry (November 1985). National Aeronautics and Space Administration.

- Plett, G. L. (2004). Extended Kalman filtering for battery management systems of LiPB-based HEV battery packs Part 3 . State and parameter estimation, *134*, 277–292. http://doi.org/10.1016/j.jpowsour.2004.02.033
- Lee, J., Nam, O., & Cho, B. H. (2007). Li-ion battery SOC estimation method based on the reduced order extended Kalman filtering ଋ, *174*, 9–15. http://doi.org/10.1016/j.jpowsour.2007.03.072
- Han, J., Kim, D., & Sunwoo, M. (2009). State-of-charge estimation of lead-acid batteries using an adaptive extended Kalman filter, *188*, 606–612. http://doi.org/10.1016/j.jpowsour.2008.11.143