KALMAN FILTER

Kalman filtering is an algorithm that provides estimates of some unknown variables given the measurements observed over time, accounting for statistical noise. It achieves this by combining the predicted state of the system and the latest measurement in a weighted average. These weights are assigned based on the relative uncertainty of the values, with more confidence placed in measurements with lower estimated uncertainty. If all noise is Gaussian, the Kalman filter minimises the mean square error of the estimated parameters.

The Kalman filter has two stages. It is a **recursive process, two steps of which are predict and update.**

Before we delve deep into how Kalman Filter actually works, we need to understand a few terms.

- ➢ **Variance**

The term variance refers to a statistical measurement of the spread between numbers in a data set. More specifically, variance measures how far each number in the set is from the mean (average), and thus from every other number in the set. Variance is often depicted by this symbol: σ^2. Here population variance basically means that we are exploiting all the data points provided.

$$\sigma^2 = \frac{\sum_{i=1}^{N} (x_i - \mu)^2}{N}$$

$\sigma^2 =$ population variance
$x_i =$ value of i^{th} element
$\mu =$ population mean
$N =$ population size

- ➢ **Covariance and Covariance Matrix**

Covariance is a statistical term that refers to a systematic relationship between two random variables in which a change in the other reflects a change in one variable.
The covariance value can range from -∞ to +∞, with a negative value indicating a negative relationship and a positive value indicating a positive relationship.

The greater this number, the more reliant the relationship. Positive covariance denotes a direct relationship and is represented by a positive number.

The covariance of two random variables x_1 and x_2 is

$$\begin{aligned} \operatorname{cov}(x_1, x_2) &\equiv E[(x_1 - \bar{x}_1)(x_2 - \bar{x}_2)] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x_1 - \bar{x}_1)(x_2 - \bar{x}_2) p(x_1, x_1) dx_1 dx_2 \\ &\equiv \sigma^2_{x_1 x_2} \end{aligned}$$

where p is the joint probability density function of x_1 and x_2.

A **covariance matrix** is a square matrix that illustrates the variance of dataset elements and the covariance between two datasets.

- **Important Technical Terms**

Now we will understand some more technical terms that we will require during the implementation of the algorithm.

To make things easier, let us suppose we have a radar trying to get the position and velocity of a target in 2D.

The set of target parameters [x, y, vx, vy] is known as the **System State**. The current state serves as the input for the prediction algorithm, while the algorithm's output is the future state, which includes the target parameters for the subsequent time interval.

The system of equations mentioned above is known as a **Dynamic Model or State Space Model.** The dynamic model describes the relationship between the input and output of the system. Apparently, if the target's current state and dynamic model are known, predicting the target's subsequent state can be easily accomplished.

In reality, the radar measurement is not entirely accurate. It contains random errors or uncertainties that can affect the accuracy of the predicted target state. The magnitude of the errors depends on various factors, such as radar calibration, beam width, and signal-to-noise ratio of the returned echo. The random errors or uncertainties in the radar measurement are known as **Measurement Noise.**

In addition, the target motion is not always aligned with the motion equations due to external factors like wind, air turbulence, and pilot manoeuvres. This misalignment between the motion equations and the actual target motion results in an error or uncertainty in the dynamic model, which is called **Process Noise**.

Due to the Measurement Noise and the Process Noise, the estimated target position can be far away from the actual target position. In this case, the radar might send the track beam in the wrong direction and miss the target.

In order to improve the radar's tracking accuracy, it is essential to employ a prediction algorithm that accounts for both process and measurement uncertainty. The most common tracking and prediction algorithm is the **Kalman Filter.**

GAUSSIAN FUNCTION AND BAYESIAN FILTERING

The mighty Gaussian function is given below:

$$\mathcal{N}(x_i;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} exp\{-\frac{1}{2\sigma^2}(x_i-\mu)^2\}$$

Kalman Filter is basically a **special case of Bayesian Filtering** wherein the state(s) and measured(m) variables follow **Gaussian Distribution and Addition Linearity**. After applying Gaussian Mathematics, we come up with a quite simple implementation of Kalman Filter as explained in the next section.

The update process in Kalman Filter can be thought of as **Bayesian inference**. If we define the actual position of the object we are tracking (that we don't actually know) as x and the actual measured position as m we get the following update equation:

$$\underbrace{P(x|m)}_{Posterior} = \frac{\overbrace{P(m|x)}^{Likelihood} \cdot \overbrace{P(x)}^{Prior}}{\underbrace{P(m)}_{Normalization}}$$

Bayes' theorem as used in the Kalman filter

- **Posterior**: The probability distribution of the actual location, given the measurement. This is what we want to infer using the update step.

- **Likelihood**: The probability distribution of the measurement, given the actual location. This is assumed to be a Gaussian distribution around the actual position.
- **Prior**: The probability distribution of the actual location to the best of our current knowledge prior to the measurement (a.k.a the predicted state).
- **Normalisation**: The measurement probability distribution, summed over all possible locations. In this context, this part is not calculated as we only care about updating the mean and covariance and don't care about the Gaussians' amplitudes.

We get the result of Predict Step using Chapman-Kolmogorov Equation

Chapman-Kolmogorov Equation

$$p(\mathbf{s}_k|\mathbf{m}_{k-1}) = \int p(\mathbf{s}_k|\mathbf{s}_{k-1})\ p(\mathbf{s}_{k-1}|\mathbf{m}_{k-1})d\mathbf{s}_{k-1}$$

Marginalization by integrating out nuisance/dummy variable

Combining the above two steps we get:

Predict Step

$$p(\mathbf{s}_k|\mathbf{m}_{k-1}) = \int p(\mathbf{s}_k|\mathbf{s}_{k-1})\ p(\mathbf{s}_{k-1}|\mathbf{m}_{k-1})d\mathbf{s}_{k-1}$$

Update Step

$$p(\mathbf{s}_k|\mathbf{m}_k) = \frac{p(\mathbf{m}_k|\mathbf{s}_k) \cdot p(\mathbf{s}_k|\mathbf{m}_{k-1})}{N_k}$$

From here we can deduce that the updated state will be a product of the predicted state and measured state. Because we are dealing here with Gaussian distributions, the product of such distributions will be yet another Gaussian distribution with closed equations for its mean and covariance.

STEPS INVOLVED

1. **Defining the State Space Model:**

a. The first step is to define the state variables of the system we want to estimate. These variables represent the internal state of the system that we are interested in estimating.
The state variable **x(k) is represented by a n-dimensional vector.**

b. Create a state transition model that describes how the state variables evolve over time (often represented by a linear system with some noise).

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_{k-1} + \mathbf{w}_{k-1}$$

u(k-1) is the input vector
w(k-1) represents the zero mean process noise

In matrix representation, x(k) can be written as: (excluding the process noise covariance matrix)

$$\mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \dot{x}_k \\ \dot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix}$$

Here u is taken as a null matrix for simplicity.

2. Initialization:

a. Initialise the state estimate: Start by estimating the initial state of the system. This is usually done based on prior knowledge or initial measurements.

b. Initialise the error covariance matrix: This matrix represents the uncertainty in the initial state estimate. It will be updated as the filter processes more measurements.

The **process noise** covariance matrix Q for 2-D Kalman filter can be written as:

$$\mathbf{Q} = \begin{array}{c} \\ x \\ y \\ \dot{x} \\ \dot{y} \end{array} \begin{array}{c} \begin{array}{cccc} x & y & \dot{x} & \dot{y} \end{array} \\ \begin{bmatrix} \sigma_x^2 & 0 & \sigma_x\sigma_{\dot{x}} & 0 \\ 0 & \sigma_y^2 & 0 & \sigma_y\sigma_{\dot{y}} \\ \sigma_{\dot{x}}\sigma_x & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & \sigma_{\dot{y}}\sigma_y & 0 & \sigma_{\dot{y}}^2 \end{bmatrix} \end{array}$$

3. **Prediction:**

    a. Use the state transition model mentioned in Step 1 to predict the state at the next time step **(a priori prediction)**. This step estimates the current state based on the previous state and any control inputs, if applicable.

    b. Update the error covariance matrix to account for the prediction uncertainty.

    First, the a priori state estimate P(k)- is predicted by using the state dynamic equation model that projects forward one step in time as follows:

$$\hat{\mathbf{x}}_k^- = A\hat{\mathbf{x}}_{k-1} + B\mathbf{u}_{k-1}$$

    where: x(k-1) is the previous estimated state (a posteriori state estimate).

    Next, the error covariance matrix P(k)- is predicted by:

$$\mathbf{P}_k^- = A\mathbf{P}_{k-1}A^T + \mathbf{Q}$$

    where P(k-1) is the previous estimated error covariance matrix and Q is the process noise covariance.

4. **Measurement Update (Correction):**

    a. Obtain a new measurement from the system.
    b. Calculate the **measurement residual** (difference between the actual measurement and the predicted measurement based on the state estimate). The measurement residual is

$$\mathbf{z}_k - H\hat{\mathbf{x}}_k^-$$

    z(k) is the measurement

H is the transformation matrix used to bring the state vector to the same dimension as the measurement vector

c. Compute the **Kalman gain**: This is a weight factor that balances the prediction and measurement information in the state update. It depends on the prediction uncertainty and the measurement uncertainty.

$$\mathbf{K}_k = \mathbf{P}_k^- H^T (\mathrm{H}\mathbf{P}_k^- H^T + \mathbf{R})^{-1}$$

d. Update the state estimate based on the measurement and the Kalman gain.

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{z}_k - H\hat{\mathbf{x}}_k^-)$$

e. Update the error covariance matrix to reflect the new state estimate's uncertainty.

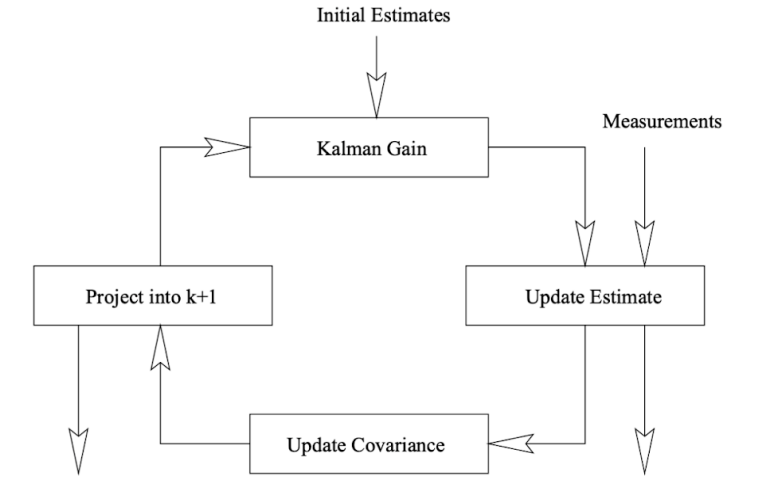
$$\mathbf{P}_k = (I - \mathbf{K}_k H)\mathbf{P}_k^-$$

5. **Repeat:**

a. Continue the prediction and measurement update steps as new measurements become available.

b. The Kalman filter will iteratively improve the state estimate by incorporating new measurements while considering the uncertainties.

The algorithm is summed up in the following flow chart.



Description	Equation
Kalman Gain	$K_k = P'_k H^T \left(H P'_k H^T + R \right)^{-1}$
Update Estimate	$\hat{x}_k = \hat{x}'_k + K_k (z_k - H \hat{x}'_k)$
Update Covariance	$P_k = (I - K_k H) P'_k$
Project into $k+1$	$\hat{x}'_{k+1} = \Phi \hat{x}_k$ $P_{k+1} = \Phi P_k \Phi^T + Q$

It's important to note that the Kalman filter assumes the **system's dynamics are linear** and that the **measurement and process noises are Gaussian and additive**. In practice, the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) may be used to handle nonlinear systems. Additionally, the filter's performance depends on the accuracy of the state space model and the accuracy of the noise models. Calibration and tuning are essential to achieve good estimation results.

OUTPUT OF CODE