Introduction to Kalman Filtering

An Engineer's Perspective

Prof. x Someone

School of Automation Science and Engineering
South China University of Technology

June 17, 2023

South China University of Technology

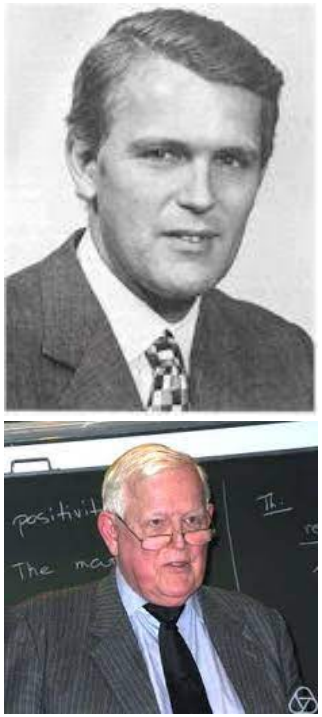
Outline

Table of Contents

South China University of Technology

Introduction to Kalman Filter

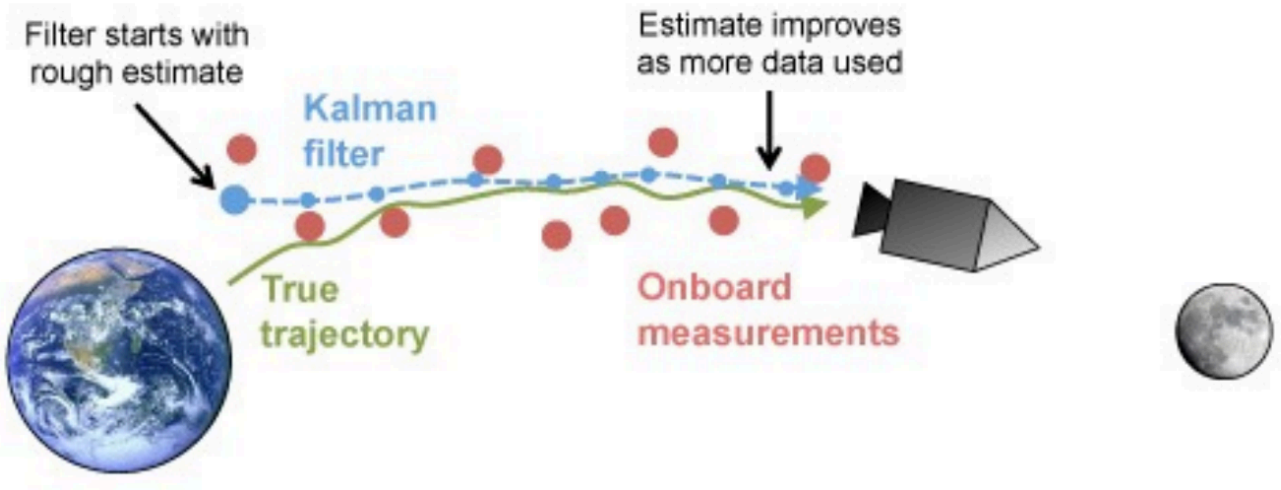
- Developed by Rudolf E. Kalman
  - Born in 1930 in Hungary
  - Education: B.S., M.S. from MIT; Ph.D. (1957) from Columbia
  - Developed Kalman Filter in 1960-61
- Filter: just a fancy word for an algorithm that takes an input (typically, a sensor signal) and calculates a function of that input
- Kalman Filter: an efficient, recursive filter that estimates the state of a dynamic system from a series of noisy measurements



South China University of Technology

Introduction to Kalman Filter

Rudolf Kalman developed the Kalman Filter in the 1960's. The filter was used to help control spacecraft navigation systems. It was even used in the NASA Apollo missions to the moon!



What is a Kalman Filter used for?

Broadly, it's useful for any type of tracking application, such as

- Tracking missiles
- Estimating position of aircraft
- Surveillance of highway traffic
- GPS-based motion estimation
- Economics applications (e.g., estimating demand for international reserves)
- Mobile robot localization!

Table of Contents

Why Use Kalman Filters?

Remark

A Kalman filter is an optimal estimation algorithm used to estimate states of a system from indirect and uncertain measurements.

Kalman filter is used when:

- The variables of interest can only be measured indirectly.
- Measurements are available from various sensors but might be subject to noise.

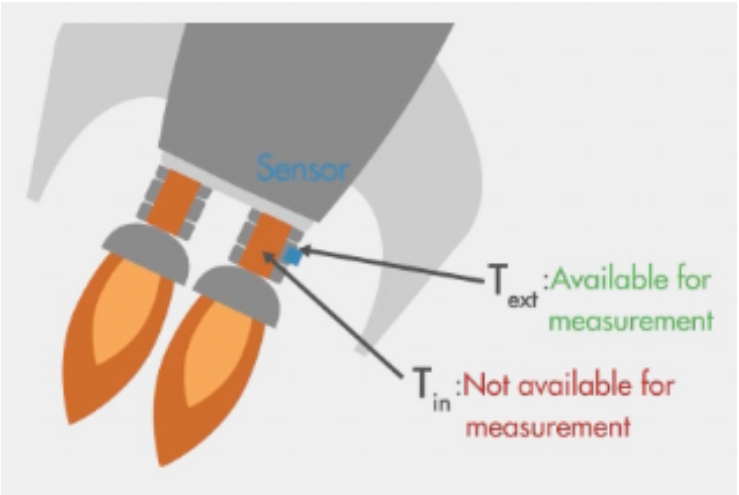


Figure: Kalman filter used to estimate the internal temperature of a combustion chamber

South China University of Technology

Why Use Kalman Filters?

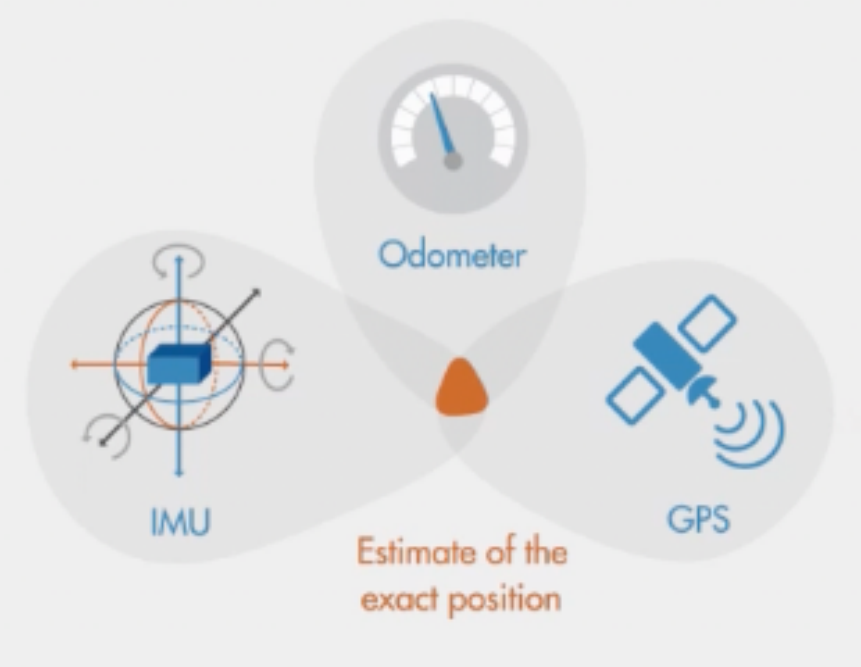


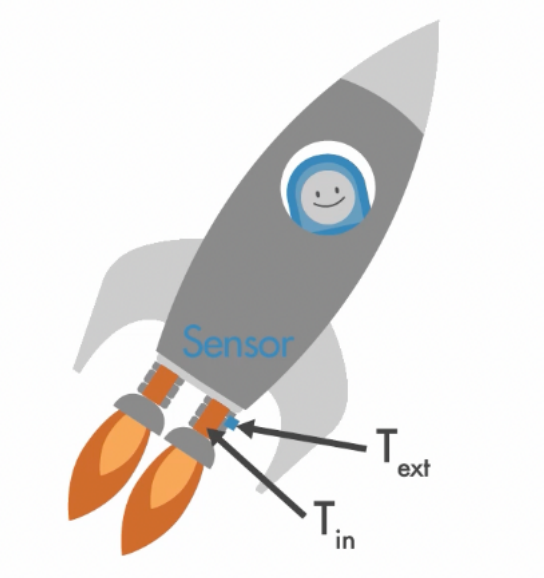
Figure: Estimate the state of a system (e.g., the position of a car) by fusing measurements from multiple sources (e.g., an inertial measurement unit (IMU), an odometer, and a GPS receiver) in the presence of noisy measurements.

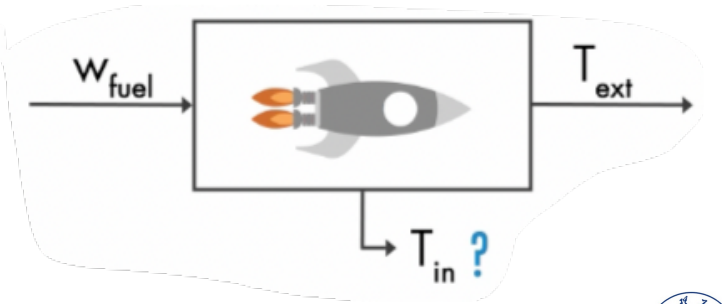
Table of Contents

South China University of Technology

How to estimate the internal temperature of a jet engine?

- There isn't any feasible way of measuring the internal temperature T_{in}, since T_{in} is too high.
- Place the sensor on a colder surface and measure the temperature there, which called external temperature T_{ext}.
- Available signals are fuel flow and external temperature measurements



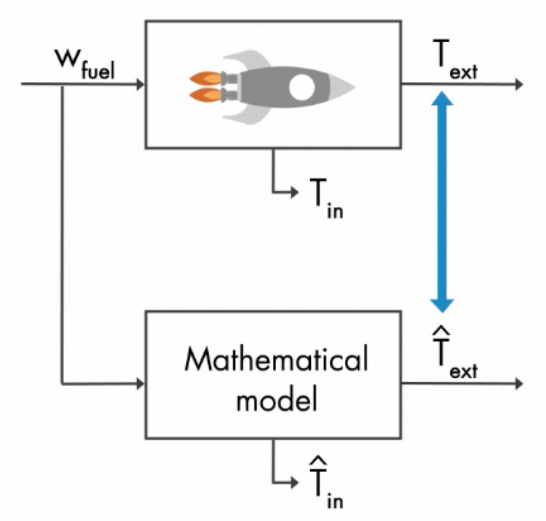


A possible solution

Remark

If the state x is shown with a hat $\hat{x}$, then it is an estimated state.

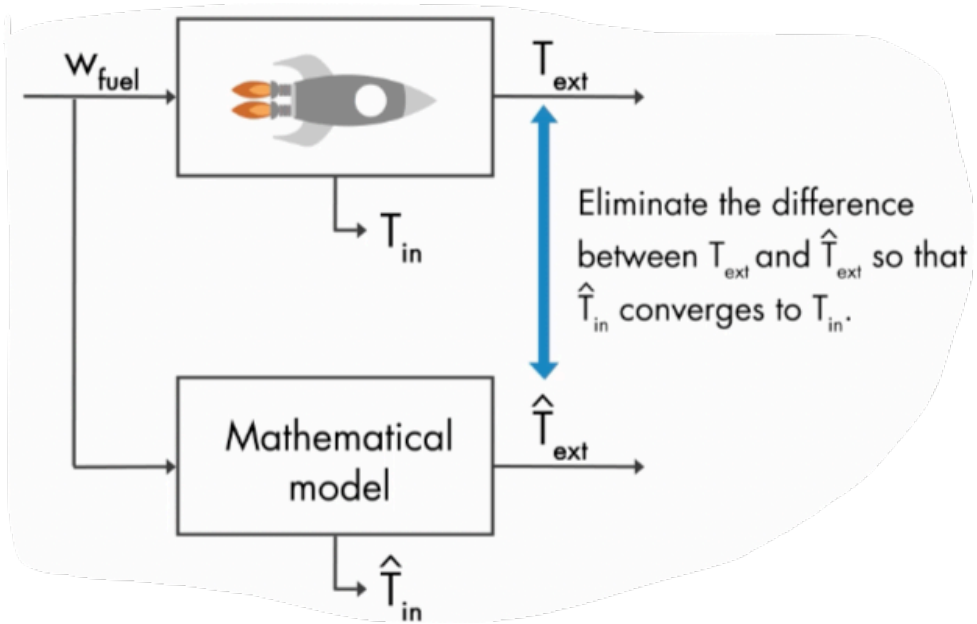
Find a mathematical model of the temperature change and, given the known inputs, calculate the internal state of the system.



- There are modeling errors
- The system is affected by uncertain factors
- A state estimator is required to estimate the internal state

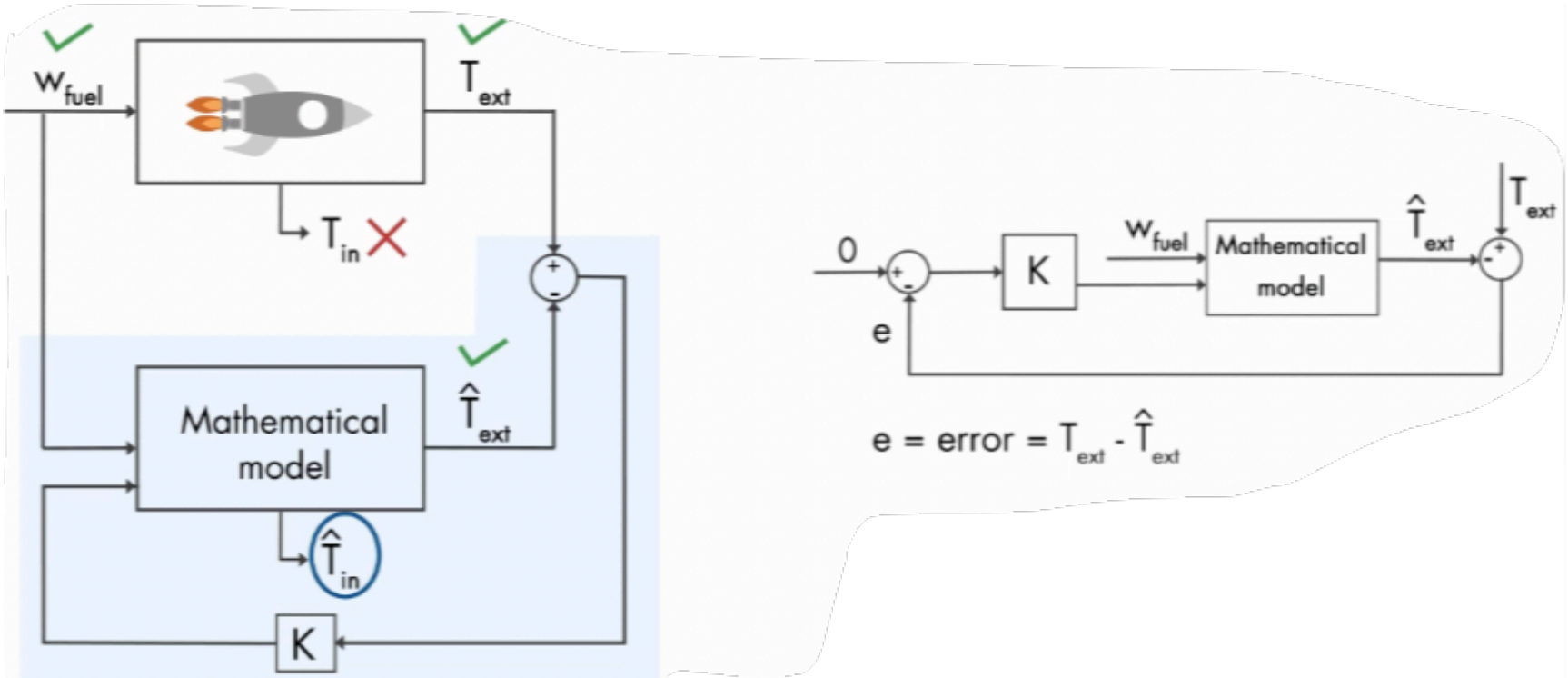
How a state estimator works?

- The goal is to match $\hat{T}_{ext}$ with the measured T_{ext}
- If $\hat{T}_{ext} = T_{ext}$, then the model will converge to the real system, and $\hat{T}_{in}$ will converge to its true value
- Minimize the difference between the estimated and measured external temperature



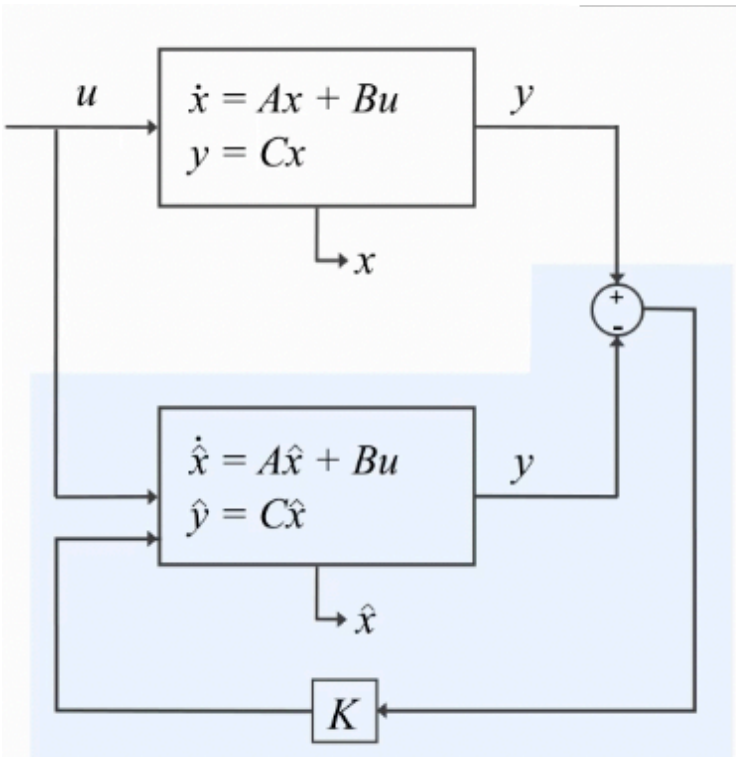
Introducing a Feedback Control System

Question: How to choose the controller gain k such that the error between the measured and estimated external temperature is minimized optimally?



mathematical perspective

We can interpret the state observer mathematically as:



$$e_{obs} = x - \hat{x}$$

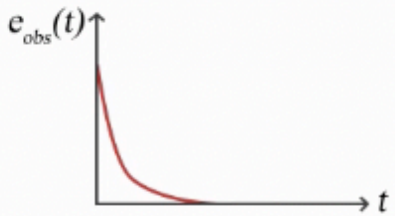
$$\dot{x} = Ax + Bu \qquad y = Cx$$

$$\dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y}) \qquad \hat{y} = C\hat{x}$$

$$\dot{e}_{obs} = (A - KC)e_{obs} \qquad y - \hat{y} = Ce_{obs}$$

$$\hookrightarrow e_{obs}(t) = e^{(A-KC)t}e_{obs}(0)$$

If $(A\text{-}KC) < 0$, then $e_{obs} \to 0$ as $t \to \infty$. So, $\hat{x} \to x$.



mathematical perspective

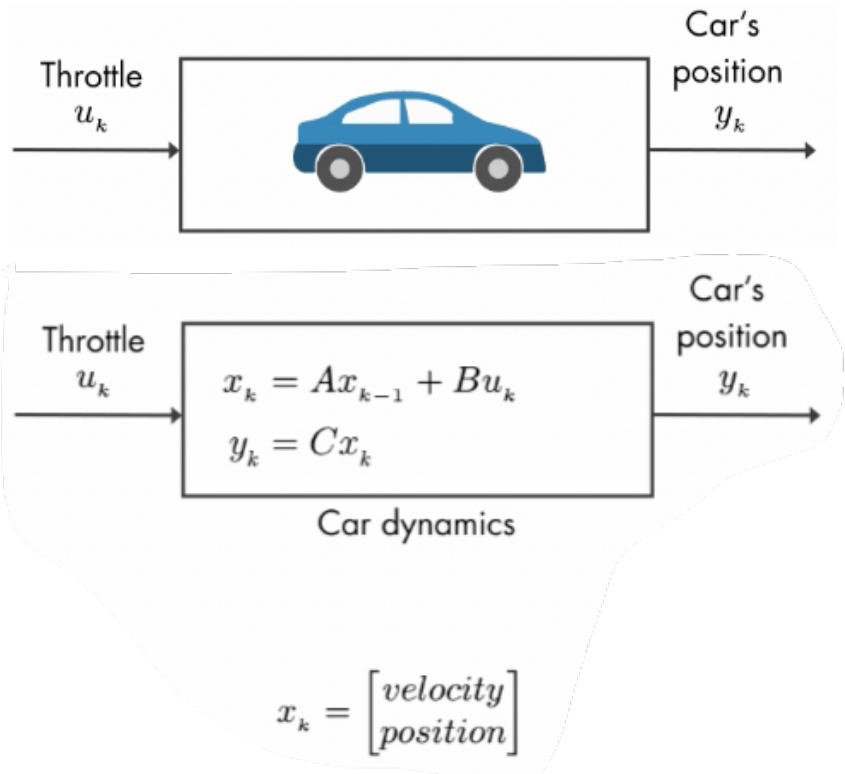
- The significance of having a feedback loop around the observer is that we can control the decaying rate of the error function by selecting the controller gain k accordingly.
- If there are some uncertainties in the mathematical model, you can't control how quickly the error will vanish.
- An optimal way of choosing the gain k is performed through the use of Kalman filters.

Table of Contents

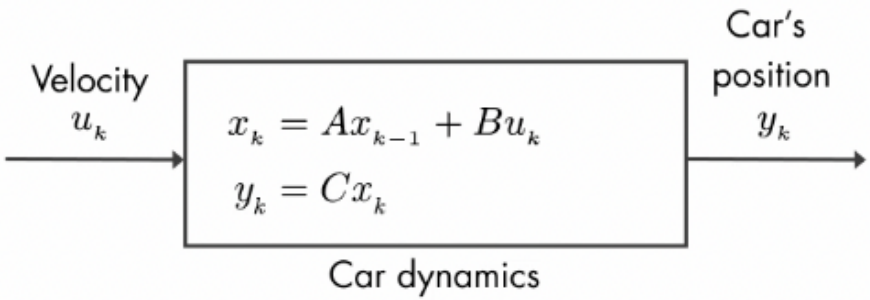
Example of estimate the car's position

Let's assume the car measures its position using GPS.

- The input to the car is a throttle.
- The output that we're interested in is the car's position.
- For such a system, we would have multiple states.



For simplicity, we take the speed of the car as input without loss of generality

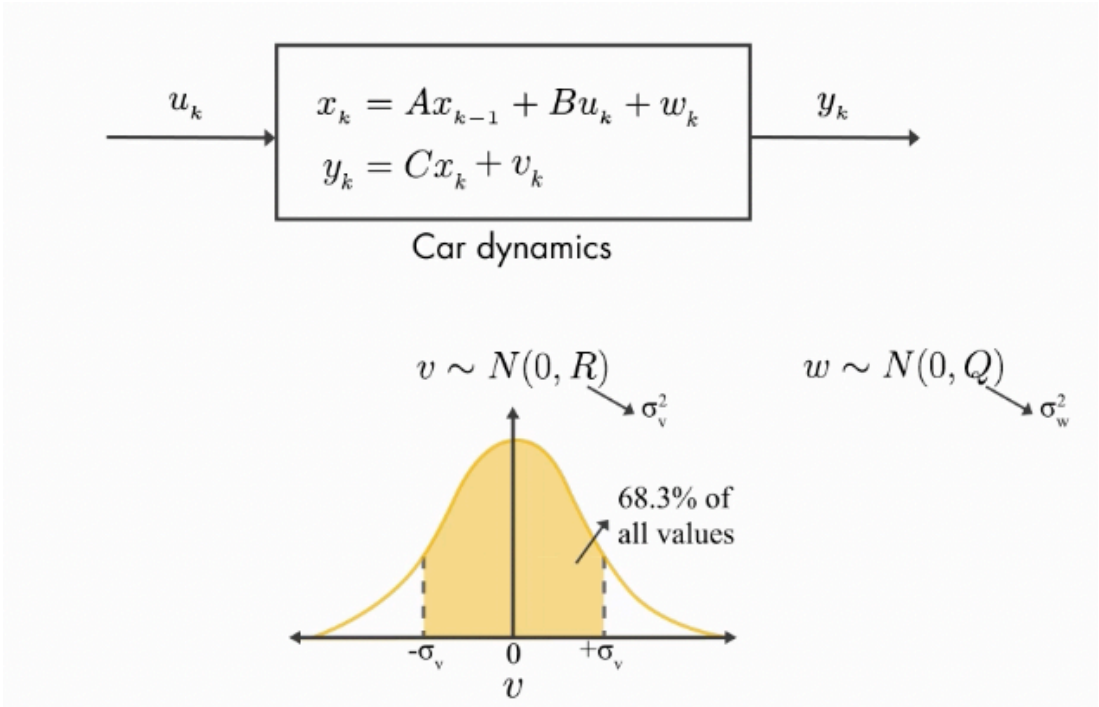


$$x_k = [position]$$

$$C = 1$$

There is measurement noise in the position measurements obtained using GPS.

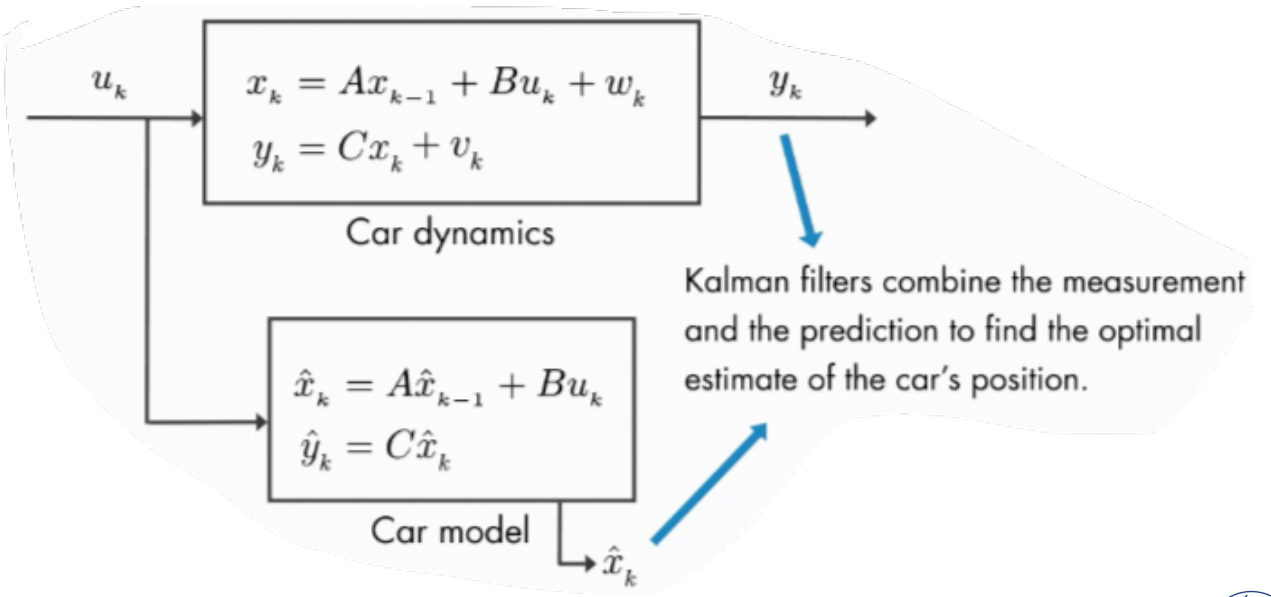
- v_k is assumed to be drawn from a Gaussian distribution with zero mean and covariance R.
- Similarly, the process noise ω_k is also random and assumes a Gaussian distribution with covariance Q.



The role of the Kalman filter

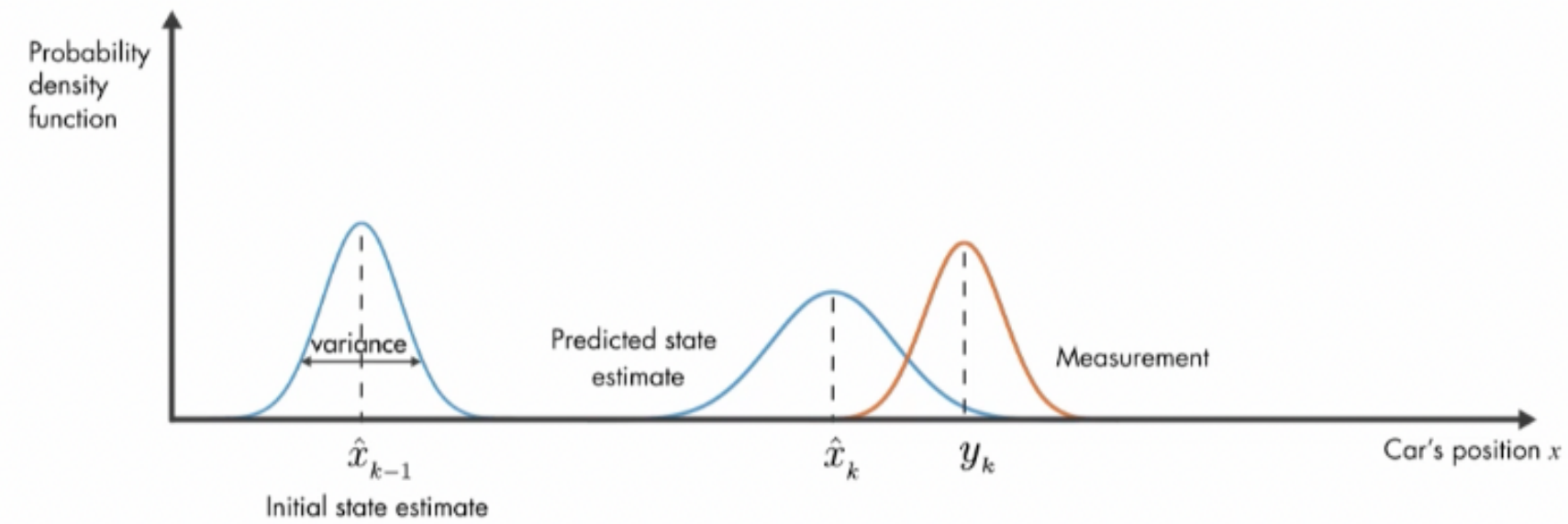
- Due to the presence of measurement noise, the measurements do not fully reflect the real position of the car.
- There is also uncertainty in the prediction of the car's position due to process noise.

This is where the Kalman filter comes into play:



Probability Perspective

We can intuitively understand how the Kalman filter works with the help of the probability density function.



Probability Perspective

It turns out that the optimal way to estimate the car's position is by combining these two pieces of information.

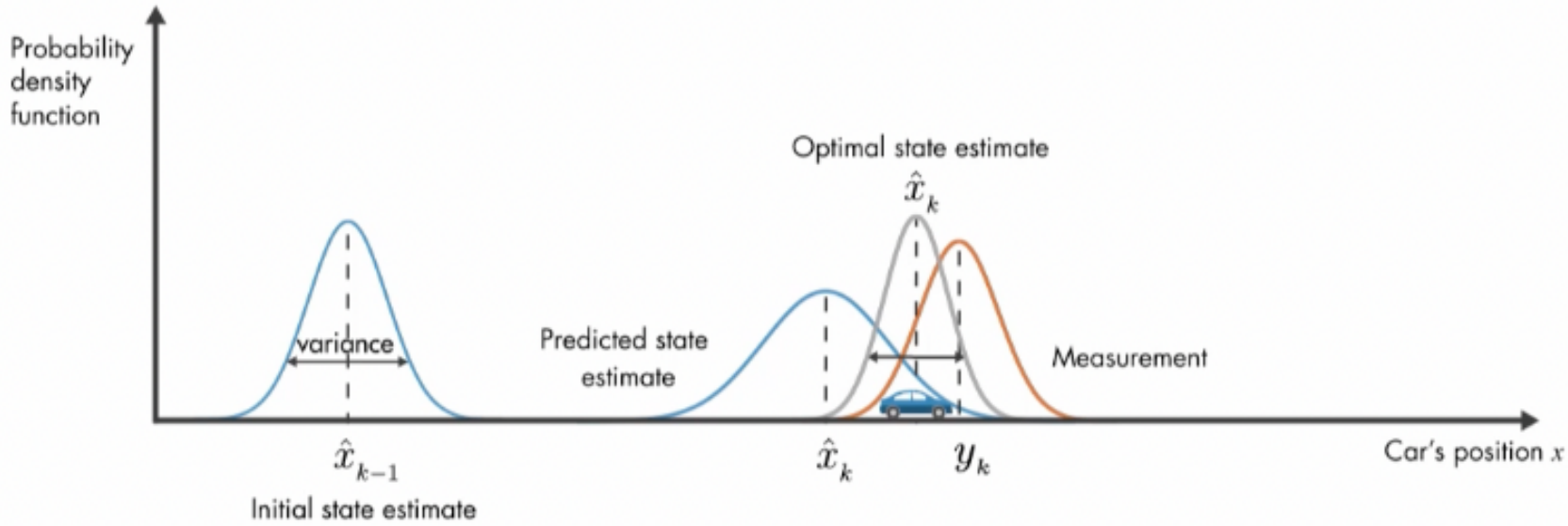
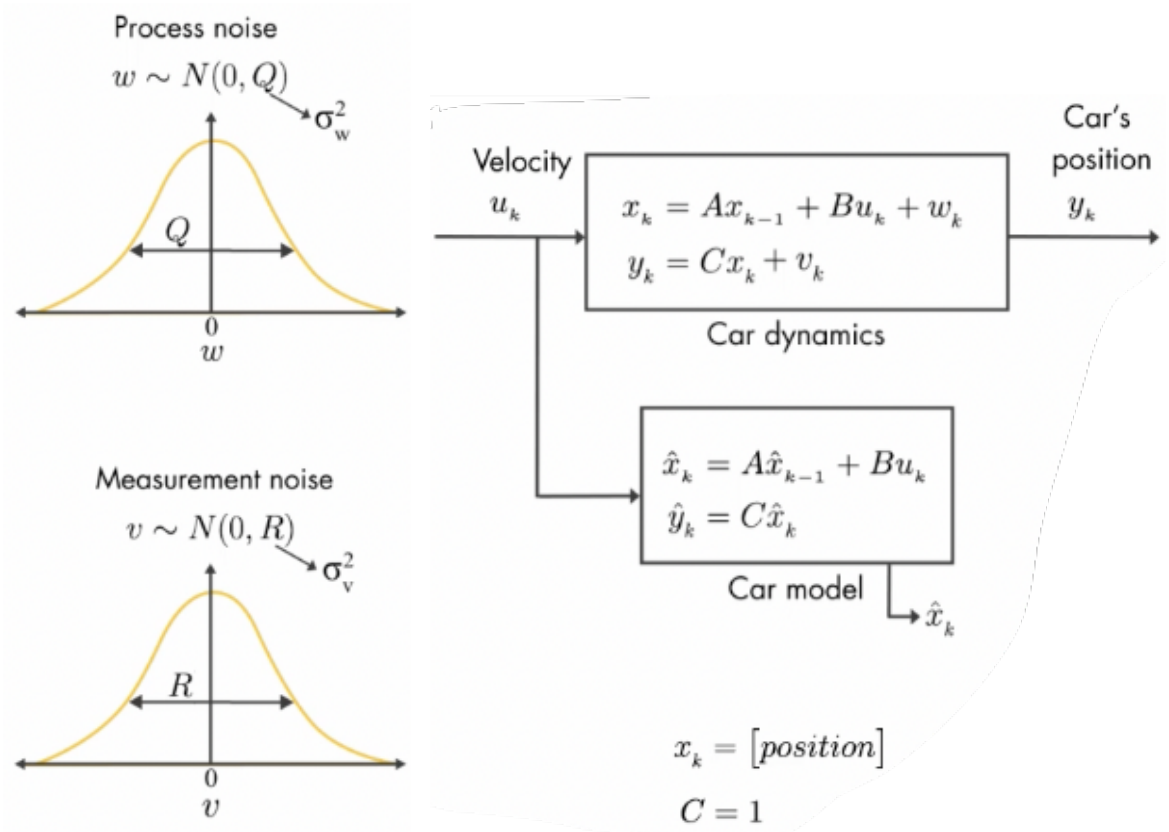


Table of Contents

Problem definition

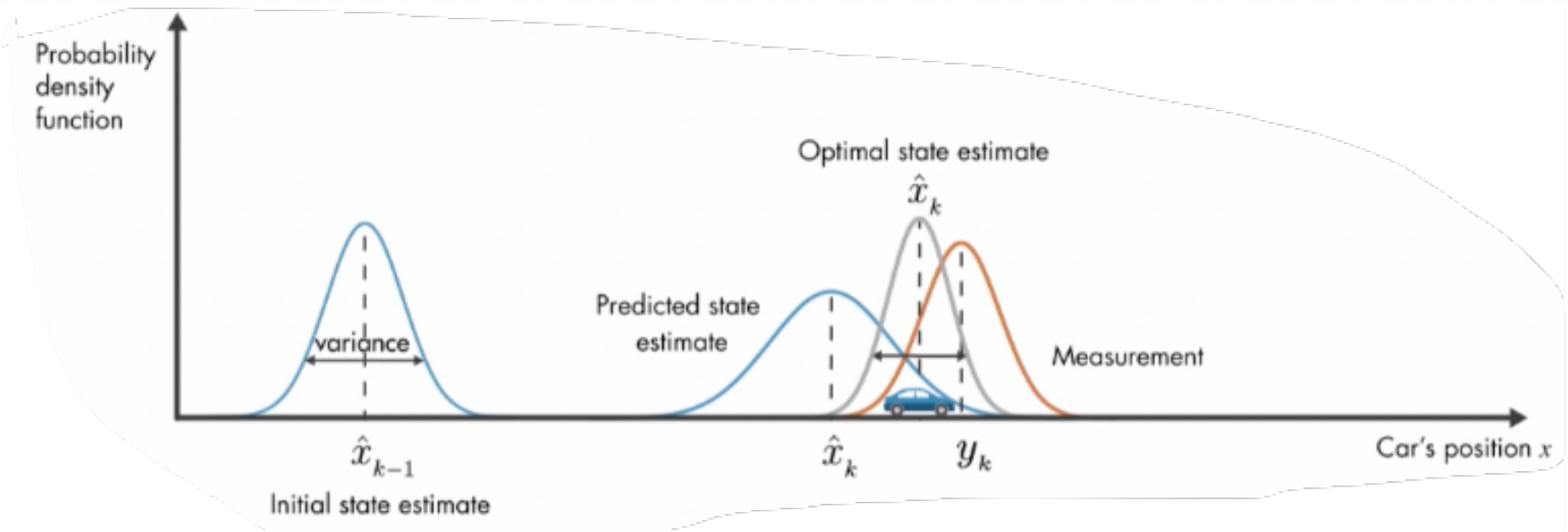
Kalman filters are used to estimate states based on linear dynamical systems in state space format. As shown in the previous example:



Kalman filter

Now we directly give the expression of the optimal estimate given by the kalman filter:

Kalman filter $\hat{x}_k = A\hat{x}_{k-1} + Bu_k + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k))$



Kalman filter

It looks like the state observer equation discussed earlier.

State observer	$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + K(y_k - C\hat{x}_k)$	Deterministic system
Kalman filter	$\hat{x}_k = A\hat{x}_{k-1} + Bu_k + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k))$	Stochastic system

Remark

Actually, a Kalman filter is a type of state observer, but it is designed for stochastic systems.

Kalman filter

The first part predicts the current state by using state estimates from the previous timestep and the current input, called the a priori estimate.

$$\hat{x}_k = \underbrace{A\hat{x}_{k-1} + Bu_k}_{\hat{x}_k^- : \text{A Priori Estimate}} + K_k(y_k - C(A\hat{x}_{k-1} + Bu_k))$$

We can now rewrite the equation, and the second part of the equation uses the measurement and incorporate it into the prediction to update the a priori estimate.

$$\overbrace{\hat{x}_k}^{\text{A Posteriori Estimate}} = \underbrace{\hat{x}_k^-}_{\text{Predict}} + \underbrace{K_k(y_k - C\hat{x}_k^-)}_{\text{Update}}$$

Kalman filter

Kalman filter algorithm consists of two stages: prediction and update.

Remark

Note that the terms "prediction" and "update" are often called "propagation" and "correction," respectively, in different literature.

The Kalman filter algorithm is summarized as follows:

Prediction

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$P_k^- = AP_{k-1}A^T + Q$$

Update

$$K_k = \frac{P_k^- C^T}{CP_k^- C^T + R}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$$

$$P_k = (I - K_k C)P_k^-$$

Kalman filter: Prediction

Remark

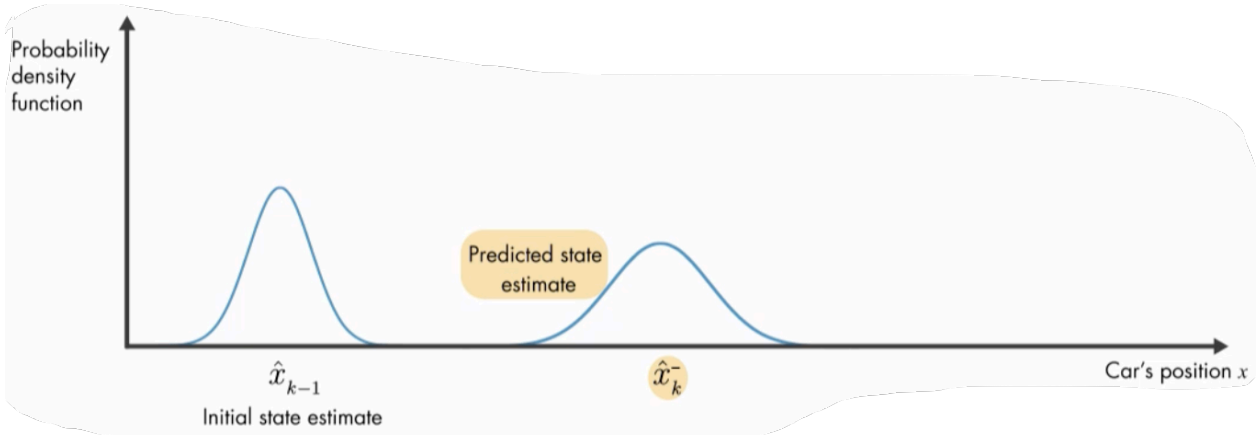
The new term P is called state error covariance. It encrypts the error covariance that the filter thinks the estimate error has.

- The system model is used to calculate the a priori state estimate:

Prediction

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$P_k^- = AP_{k-1}A^T + Q$$



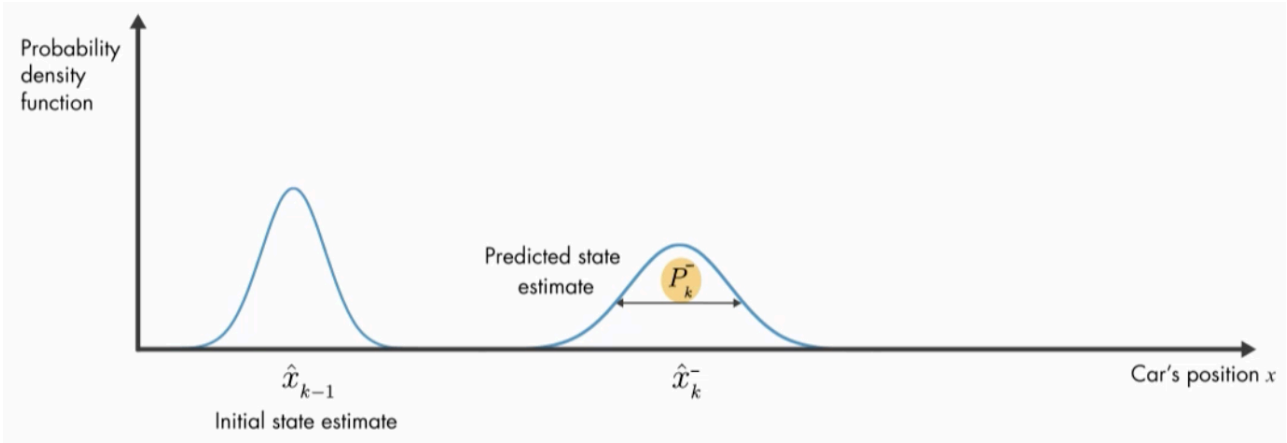
Kalman filter: Prediction

and the error covariance P_k^- .

Prediction

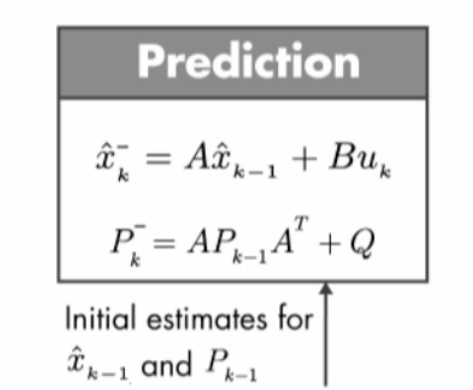
$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$$

$$P_k^- = AP_{k-1}A^T + Q$$



Kalman filter: Prediction

- P_k^- is the variance of the a priori estimate, and it can be thought of as a measure of uncertainty in the estimated state. This variance comes from the process noise and propagation of the uncertain $\hat{x}_{k-1}$.
- At the very start of the algorithm, $\hat{x}_{k-1}$ and P_{k-1} come from their initial estimates.



Kalman filter: Update

Remark

The derivation of K_k and P_k can refer to wikipedia.

- The second step of the algorithm uses the a priori estimates calculated in the prediction step and updates them to find the a posteriori estimates of the state and error covariance.
- The Kalman gain is calculated such that it minimizes the a posteriori error covariance P_k.

Update

$$K_k = \frac{\bar{P}_k C^T}{C\bar{P}_k C^T + R}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-)$$

$$P_k = (I - K_k C)\bar{P}_k$$

Kalman gain

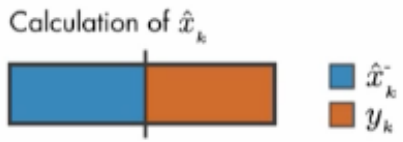
Let this bar represent the calculation of $\hat{x}_k$. By weighing the correction term, the Kalman gain determines how heavily the measurements and the a priori estimates contributes to the calculation of $\hat{x}_k$.

Update

$$K_k = \frac{\bar{P}_k C^T}{C\bar{P}_k C^T + R}$$

$$\hat{x}_k = \hat{\bar{x}}_k + K_k(y_k - C\hat{\bar{x}}_k)$$

$$P_k = (I - K_k C)\bar{P}_k$$



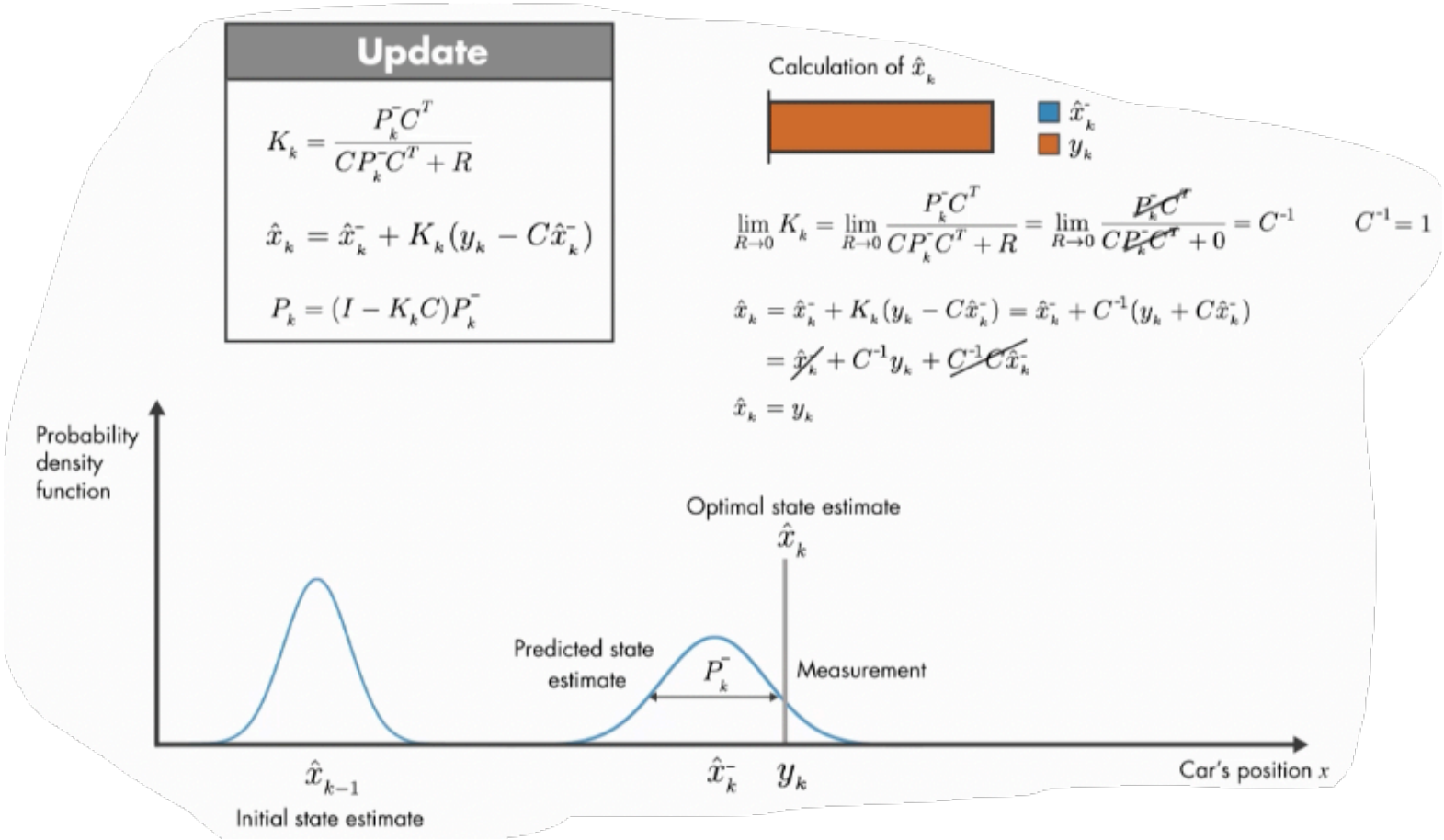
Kalman gain

Remark

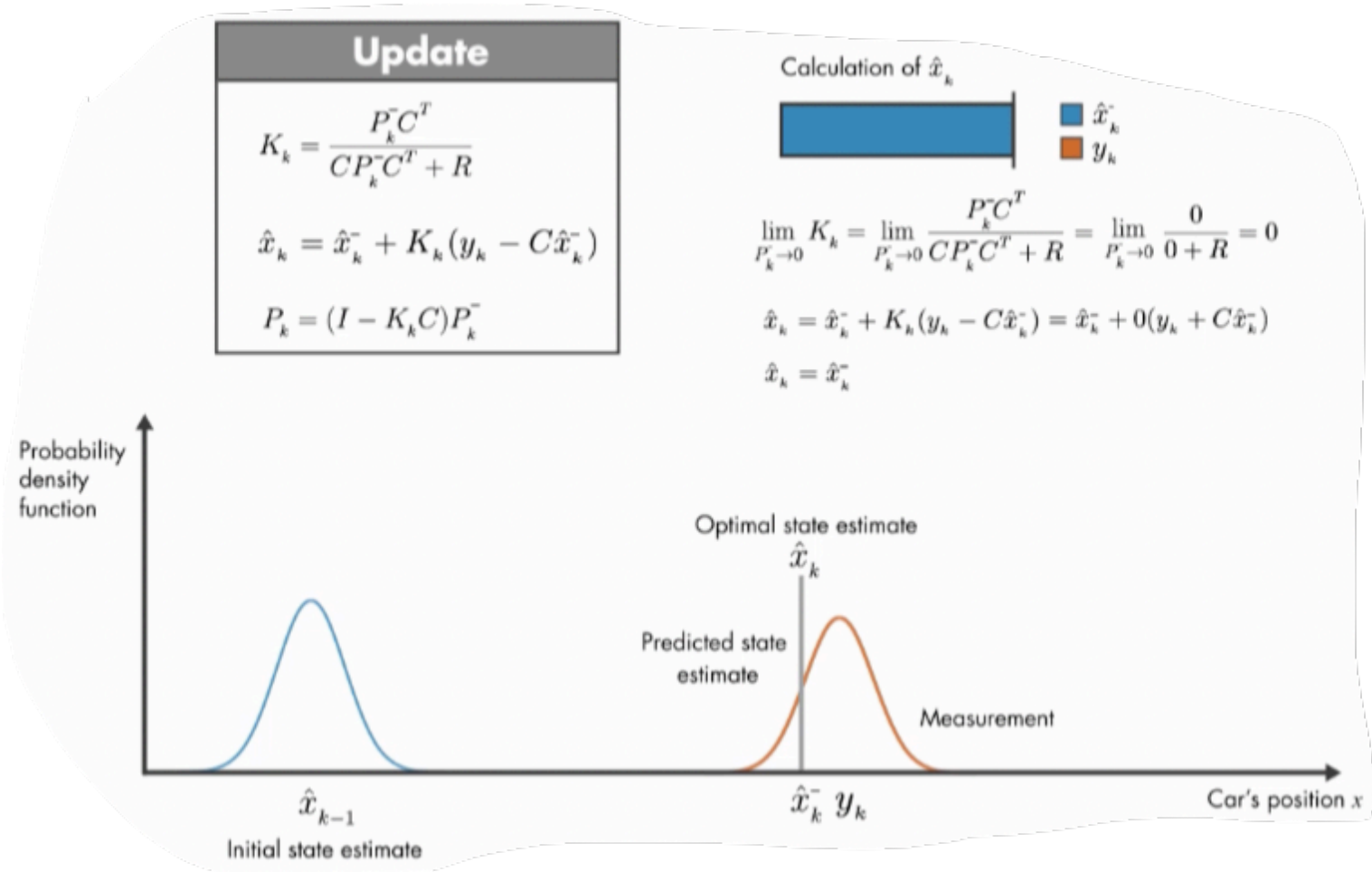
- If the measurement noise is small, the measurements is trusted more and contributes to the calculation of $\hat{x}_k$ more than the a priori state estimate does.
- In the opposite case, where the error in the a priori estimate is small, the a priori estimate is trusted more. And the computation of $\hat{x}_k$ mostly comes from this estimate.

We can also show this mathematically by looking at two extreme cases: $R \to 0$ and $P_k^- \to 0$.

If $R \to 0$, we found that the a posteriori estimate is equal to the measurement.



If $P_k^- \to 0$, the computation of $\hat{x}_k$ comes from the a priori state estimates.

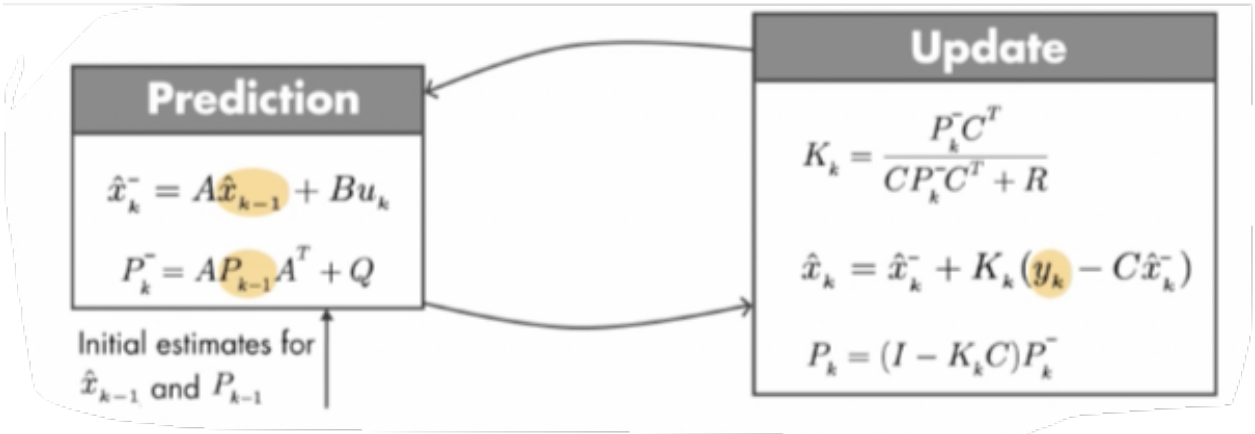


Kalman filter

Remark

Notice that to estimate the current state, the algorithm only needs the estimated state and error covariance matrix from the previous timestep and the current measurement.

Kalman filter recursive:



Kalman filter: Sensor fusion

Remark

Note that the Kalman filter is also referred as a sensor fusion algorithm.

If using additional sensor such as an IMU. the dimensions of y, C, and K matrices will change as shown here

$$\hat{x}_{k\,[1x1]} = \hat{x}^{-}_{k\,[1x1]} + K_{k\,[1x2]}(y_{k\,[2x1]} - C_{[2x1]}\hat{x}^{-}_{k\,[1x1]})$$

Kalman filter: Sensor fusion

In probability perspective, we'll be multiplying three probability density functions together to find the optimal estimate of the car's position.

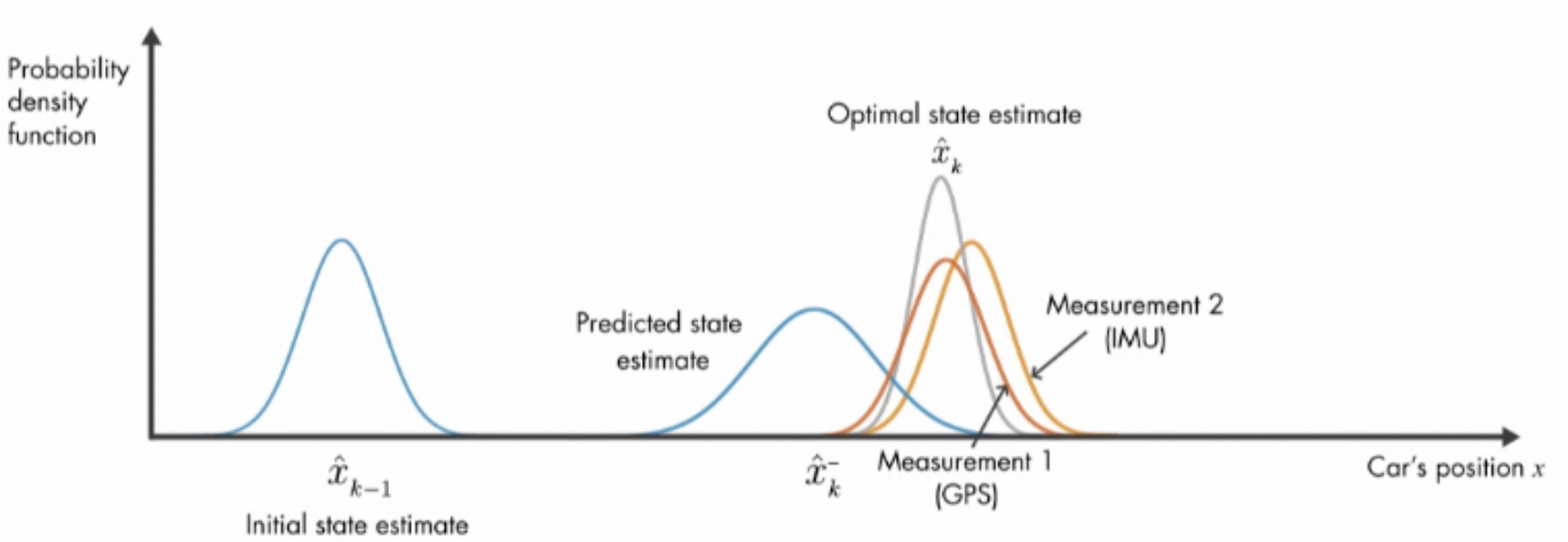


Table of Contents

In a general system, either the state transition function, or the measurement function or both may be nonlinear.

$$x_k = f(x_{k-1}, u_k) + w_k$$
$$y_k = g(x_k) + v_k$$

Nonlinear functions

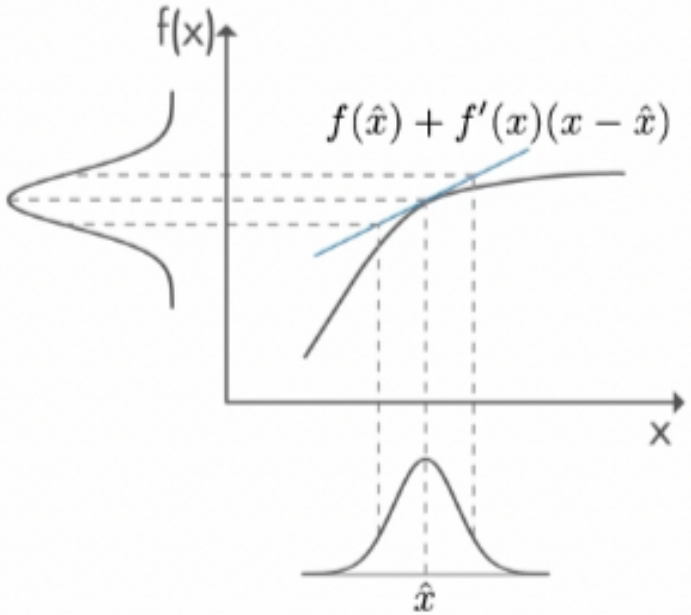
For all these cases, we need to use a nonlinear state estimator instead of a Kalman filter, as Kalman filters are only defined for linear systems:

$$x_k = Ax_{k-1} + Bu_k + w_k$$
$$y_k = Cx_k + v_k$$

In this case, we can implement an extended Kalman filter (EKF), which linearizes the nonlinear function around the mean of the current state estimate.

Extended Kalman Filters

Nonlinear transformation



System:

$$x_k = f(x_{k-1}, u_k) + w_k$$
$$y_k = g(x_k) + v_k$$

Jacobians:

$$F = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}, u_k}$$
$$G = \left.\frac{\partial g}{\partial x}\right|_{\hat{x}_k}$$

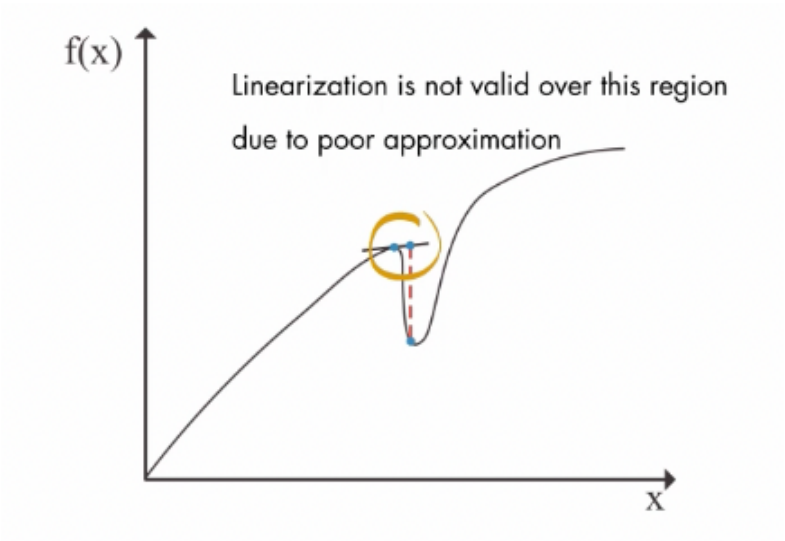
Linearized system:

$$\Delta x_k \approx F \Delta x_{k-1} + w_k$$
$$\Delta y_k \approx G \Delta x_k + v_k$$

Drawbacks to Using Extended Kalman Filters (EKFs):

- It is difficult to calculate the Jacobians (if they need to be found analytically)
- There is a high computational cost (if the Jacobians can be found numerically)
- EKF only works on systems that have a differentiable model
- EKF is not optimal if the system is highly nonlinear

For this case, maybe the unscented Kalman filter (UKF) or particle filter (PF) is useful.



UKF

Table of Contents

Modeling

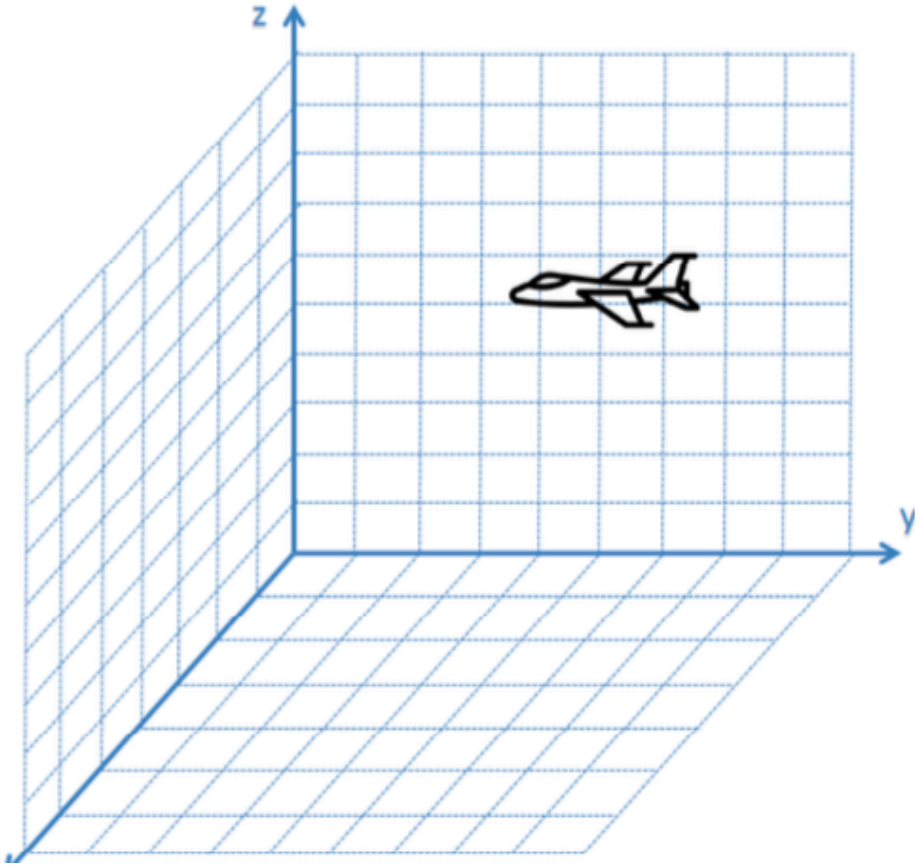
An example for implementing the Kalman filter is navigation where the vehicle state, position, and velocity are estimated by using sensor output from an inertial measurement unit (IMU) and a global navigation satellite system (GNSS) receiver. In this example, we consider only position and velocity, omitting attitude information. The three-dimensional position and velocity comprise the state vector:

$$\boldsymbol{x} = \left[\boldsymbol{p}^T, \boldsymbol{v}^T\right]^T$$

where $\boldsymbol{p} = [p_x, p_y, p_z]^T$ is the position vector and $\boldsymbol{v} = [v_x, v_y, v_z]^T$ is the velocity vector whose elements are defined in $\mathrm{x}, \mathrm{y}, \mathrm{z}$ axes.

The state in time k can be predicted by the previous state in time $k-1$ as:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{k-1} + \boldsymbol{v}_{k-1}\Delta t + \frac{1}{2}\widetilde{\boldsymbol{a}}_{k-1}\Delta t^2 \\ \boldsymbol{v}_{k-1} + \widetilde{\boldsymbol{a}}_{k-1}\Delta t \end{bmatrix}$$



where $\tilde{\boldsymbol{a}}_{k-1}$ is the acceleration applied to the vehicle. The above equation can be rearranged as:

$$\boldsymbol{x}_k = \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t \\ 0_{3\times3} & I_{3\times3} \end{bmatrix} \boldsymbol{x}_{k-1} + \begin{bmatrix} \frac{1}{2} I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix} \widetilde{\boldsymbol{a}}_{k-1}$$

where $I_{3\times3}$ and $0_{3\times3}$ denote 3×3 identity and zero matrices, respectively. The process noise comes from the accelerometer output, $\boldsymbol{a}_{k-1} = \widetilde{\boldsymbol{a}}_{k-1} + \boldsymbol{e}_{k-1}$, where $\boldsymbol{e}_{k-1}$ denotes the noise of the accelerometer output.

Suppose $\boldsymbol{e}_{k-1} \sim \mathcal{N}\left(0, I_{3\times3}\sigma_e^2\right)$. From the covariance relationship, $\mathrm{Cov}(A\boldsymbol{x}) = A\Sigma A^T$ where $\mathrm{Cov}(\boldsymbol{x}) = \Sigma$, we get the covariance matrix of the process noise as:

$$Q = \begin{bmatrix} \frac{1}{2}I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix} I_{3\times3}\sigma_e^2 \begin{bmatrix} \frac{1}{2}I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix}^T = \begin{bmatrix} \frac{1}{4}I_{3\times3}\Delta t^4 & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}\Delta t^2 \end{bmatrix} \sigma_e^2$$

Now, we have the process model as:

$$\boldsymbol{x}_k = F\boldsymbol{x}_{k-1} + B\boldsymbol{a}_{k-1} + \boldsymbol{w}_{k-1}$$

where

$$F = \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t \\ 0_{3\times3} & I_{3\times3} \end{bmatrix}$$

$$B = \begin{bmatrix} \frac{1}{2}I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix}$$

$$\boldsymbol{w}_{k-1} \sim \mathcal{N}(0, Q)$$

The GNSS receiver provides position and velocity measurements corrupted by measurement noise ν_k as:

$$\mathbf{z}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \end{bmatrix} + \boldsymbol{\nu}_k$$

It is straightforward to derive the measurement model as:

$$\mathbf{z}_k = H\boldsymbol{x}_k + \nu_k$$

where

$$H = I_{6\times 6}$$
$$\nu_k \sim \mathcal{N}(0, R)$$

Simulation settings

- In order to conduct a simulation to see how it works, let us consider $N = 20$ time steps ($k = 1, 2, 3, \ldots, N$) with $\Delta t = 1$. It is recommended to generate a time history of true state, or a true trajectory, first.
- The most convenient way is to generate the series of true accelerations over time and integrate them to get true velocity and position.
- In this example, the true acceleration is set to zero and the vehicle is moving with a constant velocity, $v_k = [5, 5, 0]^T$ for all $k = 1, 2, 3, \ldots, N$, from the initial position, $\boldsymbol{p}_0 = [0, 0, 0]$.

Simulation settings

- We need to generate noise of acceleration output and GNSS measurements for every time step. Suppose the acceleration output, GNSS position, and GNSS velocity are corrupted with noise with variances of $0.3^2, 3^2$, and 0.03^2, respectively.
- The process noise covariance matrix, Q, and measurement noise covariance matrix, R, can be constructed following the real noise statistics described above to get the best performance. However, have in mind that in real applications, we do not know the real statistics of the noises and the noises are often not Gaussian. Common practice is to conservatively set Q and R slightly larger than the expected values to get robustness.

Let us start filtering with the initial guesses

$$\hat{\boldsymbol{x}}_0^+ = [2, -2, 0, 5, 5.1, 0.1]^T$$

$$P_0^+ = \begin{bmatrix} I_{3\times 3}4^2 & 0_{3\times 3} \\ 0_{3\times 3} & I_{3\times 3}0.4^2 \end{bmatrix}$$

and noise covariance matrices

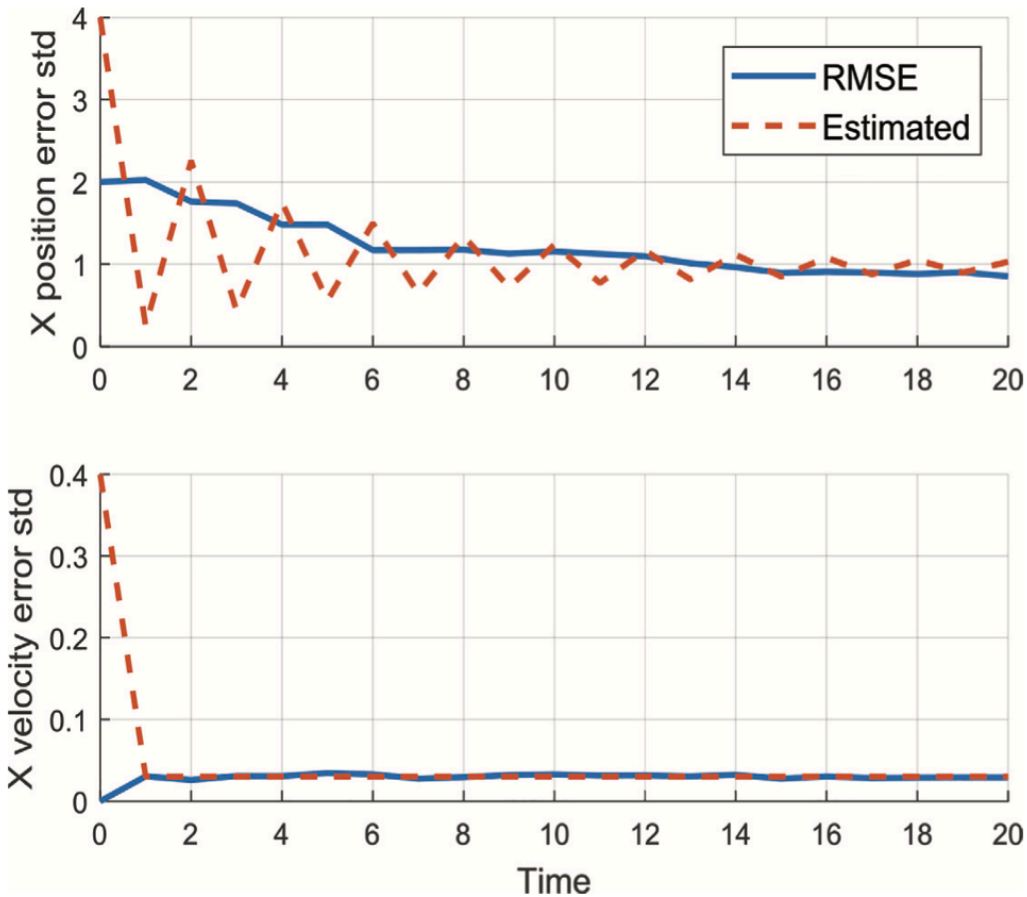
$$Q = \begin{bmatrix} \frac{1}{4}I_{3\times 3}\Delta t^4 & 0_{3\times 3} \\ 0_{3\times 3} & I_{3\times 3}\Delta t^2 \end{bmatrix} 0.3^2$$

$$R = \begin{bmatrix} I_{3\times 3}3^2 & 0_{3\times 3} \\ 0_{3\times 3} & I_{3\times 3}0.03^2 \end{bmatrix}$$

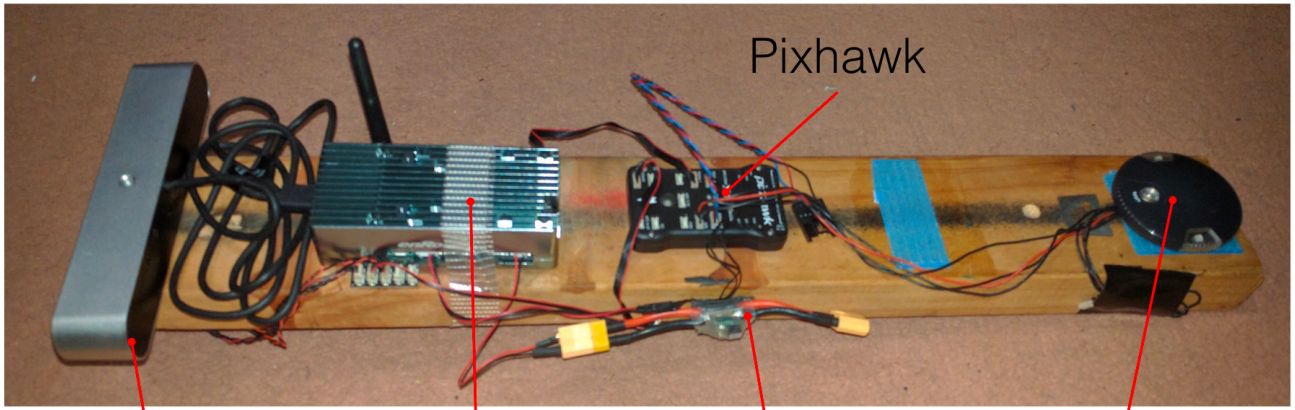
where Q and R are constant for every time step. The more uncertain your initial guess for the state is, the larger the initial error covariance should be.

Result

Actual and estimated standard deviation for x-axis estimate errors:



Hardware test configuration



Source code

- PX4: PX4 is the Professional Autopilot. Developed by world-class developers from industry and academia, and supported by an active world wide community, it powers all kinds of vehicles from racing and cargo drones through to ground vehicles and submersibles.
- ECL: Very lightweight Estimation and Control Library. Added support for running offline by Meng ChaoHeng.