An Implementation Guide on Kalman Filter

Author: Jingxue Jiang

Sammendrag legges inn her. Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

May 20, 2024

Jingxue Jiang
Queen's University, Canada
jingxue07@gmail.com

For more information please visit
www.jiangcv.com

Chapter 1 Kalman Filter

1.1 Equations of the Kalman Filter

1.1.1 State Prediction Equation

This equation works by taking the current state of the system (obtained at time step k-1) along with the current control actions (applied at time step k) and predicting the state of the system at the next time step.

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{F}_k \, \hat{\boldsymbol{x}}_{k-1} + \boldsymbol{G}_k \, \boldsymbol{u}_k \tag{1.1}$$

Where:

- $\hat{\boldsymbol{x}}_k^-$ is a predicted system state vector at time step k before correction
- $\hat{\boldsymbol{x}}_{k-1}$ is an estimated system state vector obtained at time step k-1
- $\boldsymbol{u}_k$ is a control vector - the known control inputs to the system
- $\boldsymbol{F}_k$ is a state transition matrix - describes how the system's state vector evolves from one time step to the next
- $\boldsymbol{G}_k$ is a control matrix - also called input transition matrix - describes how control actions affects the system's state

State Prediction Equation Dimensions

n is the dimension of the state vector $\hat{\boldsymbol{x}}_k$, and m is the dimension of the control vector $\boldsymbol{u}_k$.

$\hat{\boldsymbol{x}}_k^- \in \mathbb{R}^n$

$\boldsymbol{F}_k \in \mathbb{R}^{n \times n}$

$\boldsymbol{u}_k \in \mathbb{R}^m$

$\boldsymbol{G}_k \in \mathbb{R}^{n \times m}$

System State Vector $\hat{x}$

The system's state is any quantity of interest for that system such as position, velocity, and acceleration. In other words, the variables in the state vector may represent attributes of the system that we wish to know.

It's useful to remember that the state can contain any number of variables, and represent anything you want.

The state vector $\hat{\boldsymbol{x}}_k$ is also named **belief**, or **mean**.

Ex: Consider a car moving in a one-dimensional space. The state vector $\hat{\boldsymbol{x}}_k$ that describes the estimated vehicle position and velocity is:

$$\hat{\boldsymbol{x}}_k = \begin{bmatrix} \hat{p}_k \\ \hat{\dot{p}}_k \end{bmatrix}$$

State Transition Matrix F

The state transition matrix is a mathematical model that describes how the system's state changes over time. In navigation applications, this matrix is generally a function of the system dynamics.

It is a matrix which allows us to **predict** our next state. For example, a arbitrary state transition matrix $\boldsymbol{F}$ could be:

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}$$

The content of the matrix $\boldsymbol{F}$ is irrelevant, just remember the state transition matrix $\boldsymbol{F}$ maps the system's state from the present to the future.

$$\begin{aligned} \hat{\boldsymbol{x}}_k &= \boldsymbol{F}\,\hat{\boldsymbol{x}}_{k-1} \\ &= \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \hat{\boldsymbol{x}}_{k-1} \end{aligned}$$

Note: the content of the state transition matrix $\boldsymbol{F}$ varies depending on the mathematical models that governs the system. The derivation of $\boldsymbol{F}$ for this particular problem is discussed in the EXAMPLE section.

Control Vector u

There might be some changes that aren't related to the state itself - the outside world could be affecting the system. For example, if the state models the motion of a car, the driver might press the brake, causing the vehicle to decelerate, this is an case where an **external influence** affecting the system's state - the car's position and velocity.

Ex: Consider a car moving in a one-dimensional space. The control vector $\boldsymbol{u}_k$ that describes the known acceleration is:

$$\boldsymbol{u}_k = a_k$$

Ex: Consider a car moving in a two-dimensional space. The control vector $\boldsymbol{u}_k$ that describes the known acceleration along the x-y axis is:

$$\boldsymbol{u}_k = \begin{bmatrix} \ddot{x}_k \\ \ddot{y}_k \end{bmatrix}$$

Control Matrix G

The control transition matrix, also named input transition matrix, maps control actions (variables in control vector) to the system's state (variables in state vector). In essence, it tells us how the system's state changes when an input action is applied.

$$\begin{aligned}\hat{\boldsymbol{x}}_k &= \boldsymbol{G}\,\boldsymbol{u_k} \\ &= \begin{bmatrix} \frac{1}{2}\Delta t \\ \Delta t \end{bmatrix} \boldsymbol{u_k}\end{aligned}$$

Putting Everything Together - Example

Insert example here

1.1.2 Eq2: Covariance Prediction Equation

This equation works by taking the current state of the system (obtained at time step k-1) along with the current control actions (applied at time step k) and predicting the state of the system at the next time step.

$$\boldsymbol{P}_k^- = \boldsymbol{F}_k\,\boldsymbol{P}_{k-1}\,\boldsymbol{F}_k^T + \boldsymbol{Q}_k \tag{1.2}$$

Where:

- $\boldsymbol{P}_k^-$ is a predicted state covariance matrix before correction/update, used to represent the uncertainty in the state estimate
- $\boldsymbol{P}_{k-1}$ is an previous estimate of the state covariance matrix
- $\boldsymbol{F}_k$ is a state transition matrix - describes how the system's state vector evolves from one time step to the next
- $\boldsymbol{Q}_k$ is a system noise covariance matrix - to account for uncertainty in the system's state transition matrix - it is a tunable parameter

Covariance Prediction Equation Dimensions

n is the dimension of the state vector $\hat{\boldsymbol{x}}_k$, aka the number of variables in the state vector.

$\boldsymbol{P_k^-} \in \mathbb{R}^{n\times n}$
$\boldsymbol{F}_k \in \mathbb{R}^{n\times n}$
$\boldsymbol{Q}_k \in \mathbb{R}^{n\times n}$

Important concepts:

- A covariance matrix is always a square matrix
- two
- three

State Covariance Matrix P

This state covariance matrix $\boldsymbol{P}$ is a measure of how accurate your state estimate is, it is how uncertain you should be in the estimated state.

System Noise Covariance Matrix Q

Also named process noise covariance matrix. The matrix $\boldsymbol{Q}$ is a measure of how accurate your model is - some dynamics are too complicated to be modelled and are assumed as process noise.

the noise covariance matrix $\boldsymbol{Q}$ will be full in general, not a diagonal, because there is correlation between the state variables. For example, suppose you are tracking position and velocity in x-axis for a car. Process noise (which Q is modelling) might be things like wind, bumps in the road, and so on. If there is a change in velocity due to this noise, there will also be a change in position. They are correlated, and so the off-diagonal elements will be non-zero for those terms.

1.1.3 Kalman Gain Computation Equation

This equation works by taking the current state of the system (obtained at time step k-1) along with the current control actions (applied at time step k) and predicting the state of the system at the next time step.

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \, \boldsymbol{H}_k^T \left(\boldsymbol{H}_k \, \boldsymbol{P}_k^- \, \boldsymbol{H}_k^T + \boldsymbol{R}_k \right)^{-1} \tag{1.3}$$

Where:

$\boldsymbol{K}_k$	is the Kalman Gain
$\boldsymbol{P}_k^-$	is the predicted state covariance matrix before correction/update,
$\boldsymbol{H}_k$	is the observation matrix - convert the system state estimate from the state space to the measurement space
$\boldsymbol{R}_k$	is the measurement noise covariance matrix

1.1.4 State Update Equation

This equation updates/corrects the the estimates of the state vector. The predicted state has been corrected by the Kalman gain times the innovation. The term $(\boldsymbol{z}_k - \boldsymbol{H}_k \, \hat{\boldsymbol{x}}_k^-)$ is called the **innovation**, it is the difference between the predicted measurement and the actual measurement.

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k \left(\boldsymbol{z}_k - \boldsymbol{H}_k \, \hat{\boldsymbol{x}}_k^- \right) \tag{1.4}$$

Where:

$\hat{\boldsymbol{x}}_k^+$ is a predicted system state vector at time step k after correction

$\hat{\boldsymbol{x}}_k^-$ is a predicted system state vector at time step k before correction

$\boldsymbol{K}_k$ is a Kalman Gain

$\boldsymbol{z}_k$ is a measurement vector - each of the measurements taken from a sensor is stored in a measurement vector

$\boldsymbol{H}_k$ is an observation matrix - convert the system state estimate from the state space to the measurement space

State Update Equation Dimensions

n is the dimension of the state vector $\hat{\boldsymbol{x}}_k$, aka the number of variables in the state vector.
z is the dimension of the measurement vector.

$\hat{\boldsymbol{x}}_k^+ \in \mathbb{R}^n$

$\hat{\boldsymbol{x}}_k^- \in \mathbb{R}^n$

$\boldsymbol{z}_k \in \mathbb{R}^z$

$\boldsymbol{H}_k \in \mathbb{R}^{z \times n}$

$\boldsymbol{K}_k \in \mathbb{R}^{n \times z}$

1.1.5 Covariance Update Equation

This equation updates/corrects the state covariance matrix.

$$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^- (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k)^T + \boldsymbol{K}_k \boldsymbol{R}_k \boldsymbol{K}_k^T \tag{1.5}$$

Where:

$\boldsymbol{P}_k^+$ is the predicted state covariance matrix after correction/update,

$\boldsymbol{P}_k^-$ is the predicted state covariance matrix before correction/update,

$\boldsymbol{K}_k$ is the Kalman Gain

$\boldsymbol{H}_k$ is the observation matrix - convert the system state estimate from the state space to the measurement space

$\boldsymbol{R}_k$ is the measurement noise covariance matrix

$\boldsymbol{I}$ is the identity matrix

Chapter 2 Next Chapter

2.1 Linearization via Taylor Series

2.1.1 Why is it important?

In the context of state estimation, linearization means given a non-linear function, we'd like to find a linear equivalent of that function, so that we can map/propagate belief (a probability distribution) through non-linear system models/dynamics.

- The reason we linearize is for the propagation of covariance, not for the mean, you can just evaluate the nonlinear function to get the mean.
- Get the Jacobian of the map
  - By hand
  - Symbolic package (Analytical solution)
  - Automatic differentiation (Numerical Solution)
- Analytic formulas for the derivatives will make it faster. if you have a closed-form evaluation of these Jacobians if the model is simple, it is very beneficial for real-time applications. If the model is too complicated, you can try use automatic differentiation
- Linearize once to get the analytical/closed-form formula, and evaluate the Jacobian every iteration of the loop because the linearization point will change.

2.1.2 Taylor Series Expansion Formula

The Taylor series expansion of a function $f(x)$ that is **infinitely differentiable** at a point x = a is given by

$$f(x) = f(a) + \frac{df}{dx}\Big|_{x=a}(x-a) + \frac{1}{2!}\frac{d^2f}{dx^2}\Big|_{x=a}(x-a)^2 + \cdots + \frac{1}{n!}\frac{d^nf}{dx^n}\Big|_{x=a}(x-a)^n$$

Where:

$f(x)$ is a real function

$f(a)$ is f(x) evaluated at x=a - the derivative of order zero of f is defined to be f itself

$\left.\frac{d^n f}{dx^n}\right|_a$ is the nth order derivative of f evaluated at the point a - equivalent to $f^n(a)$
- the **a** subscript next to the vertical bar (|) means that the derivative is evaluated at the point x = a

$\left.\frac{d^2 f}{dx^2}\right|_a$ is the second order derivative of f evaluated at the point a - equivalent to $f^2(a)$

$\left.\frac{d^f}{dx}\right|_a$ is the first order derivative of f evaluated at the point a - equivalent to $f^1(a) = f'(a)$

In practice, however, we often cannot compute the infinite Taylor series of the function, or the function is not infinitely differentiable at some points. Therefore, we often have to truncate the Taylor series (use a finite number of terms) to approximate the function.

Example of Using Taylor Series to Approximate a Function

Let's say we want to expand the function:

$$f(x) = \sin(x)$$

around the point $x = 0$ (i.e., $a = 0$).

The Taylor series expansion would be:

$$f(x) = \sin(0) + \cos(0)(x-0) + \frac{-\sin(0)}{2!}(x-0)^2 + \frac{-\cos(0)}{3!}(x-0)^3 + \cdots$$

Since $\sin(0) = 0$ and $\cos(0) = 1$, the expansion becomes:

$$f(x) = 0 + 1 \cdot x + 0 \cdot x^2 + -\frac{1}{6} \cdot x^3 + \cdots$$

Which simplifies to:

$$f(x) = x - \frac{1}{6} \cdot x^3 + \cdots$$

This is the Taylor series expansion of the sine function around $x = 0$, truncated to the third order.

Infinite differentiability

A function $f(x)$ is said to be infinitely differentiable at a point a if it can be differentiated repeatedly at that point, and each derivative exists and is finite.

In other words, the function has a first derivative $f'(a)$, a second derivative $f''(a)$, a third derivative $f'''(a)$, and so on, and each of these derivatives is a finite value.

Let $f(x) = x^2$ and let's say we want to know if it's infinitely differentiable at $a = 2$.

$$\begin{aligned} f'(x) &= 2x, & f'(2) &= 4 \text{ (finite)} \\ f''(x) &= 2, & f''(2) &= 2 \text{ (finite)} \\ f'''(x) &= 0, & f'''(2) &= 0 \text{ (finite)} \\ f''''(x) &= 0, & f''''(2) &= 0 \text{ (finite)} \end{aligned}$$

And so on. Since each derivative exists and is finite at $a = 2$, we can say that $f(x) = x^2$ is infinitely differentiable at $a = 2$.

Note that not all functions have this property. For example, the function $f(x) = |x|$ is not infinitely differentiable at $a = 0$ because its first derivative is not defined at $a = 0$.

2.1.3 First Order Taylor Series Approximation

Case 1: Scalar-valued function of single variable $f : R \to R$

The notation $f : \boldsymbol{R} \to \boldsymbol{R}$ indicates that the function f takes a real number as input and produces a real number as output. Ex. $f(x) = x^2, f(2) = 4$.

The linear approximation for $f(x)$ at an operating point x = a is given by the first-order Taylor polynomial

$$f(x) \approx f(a) + \frac{df}{dx}\Big|_a (x - a)$$

Alternatively written as

$$f(x) \approx f(a) + f'(a)(x - a) \tag{2.1}$$

Case 2: Scalar-valued function of multiple variables $f : R^n \to R$

The notation $f : \boldsymbol{R}^n \to \boldsymbol{R}$ indicates that the function f takes a n-dimensional real vector as input and produces a scalar as output. Ex. $f(\boldsymbol{x}) = f(x_1, x_2, x_3) = x_1 + x_2 + x_3, f(1, 2, 3) = 6$.

Generalizing the Taylor series expansion to scalar-valued functions of multiple variables:

$$f(\boldsymbol{x}) = f(x_1, x_2, x_3, \cdots, x_n)$$

The linear approximation for $f(\boldsymbol{x})$ at an operating point $\boldsymbol{a}$ (a vector) is:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{a}) + J_f(\boldsymbol{a})(\boldsymbol{x} - \boldsymbol{a}) \tag{2.2}$$

Where:

- $f(\boldsymbol{x})$ is a scalar-valued function of multiple variables equivalent to $f(x_1, x_2, x_3, \cdots, x_n)$
- $f(\boldsymbol{a})$ is the function f evaluated at the operating point $(a_1, a_2, a_3, \cdots, a_n)$ - a scalar
- $J_f(\boldsymbol{a})$ a matrix of partial derivatives(also the Jacobian matrix) - a $1 \times n$ row vector
- $(\boldsymbol{x} - \boldsymbol{a})$ is the first order term of the taylor series expansion
- $\boldsymbol{x}$ is a $n \times 1$ column vector (variables of f)
- $\boldsymbol{a}$ is a $n \times 1$ column vector (operating points) of size

when $f : \boldsymbol{R}^n \to \boldsymbol{R}$ is a scalar-valued function of multiple variables, the Jacobian matrix reduces to the row vector $J_f(\boldsymbol{a})$, containing all first-order partial derivatives of $f(\boldsymbol{x})$

$$J_f(\boldsymbol{a}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\boldsymbol{a}) & \frac{\partial f}{\partial x_2}(\boldsymbol{a}) & \cdots & \frac{\partial f}{\partial x_n}(\boldsymbol{a}) \end{bmatrix} \tag{2.3}$$

Example: the linear approximation to $f(x, y, z)$ about the operating point (a, b, c) is

$$\begin{aligned} f(x,y,z) &\approx f(a,b,c) + J_f(\boldsymbol{a})(\boldsymbol{x}-\boldsymbol{a}) \\ &\approx f(a,b,c) + \begin{bmatrix} \frac{\partial f}{\partial x}(\boldsymbol{a}) & \frac{\partial f}{\partial y}(\boldsymbol{a}) & \frac{\partial f}{\partial z}(\boldsymbol{a}) \end{bmatrix} \begin{bmatrix} x-a \\ y-b \\ z-c \end{bmatrix} \\ &\approx f(a,b,c) + \frac{\partial f}{\partial x}(\boldsymbol{a})(x-a) + \frac{\partial f}{\partial y}(\boldsymbol{a})(y-b) + \frac{\partial f}{\partial z}(\boldsymbol{a})(z-c) \\ &\approx f(a,b,c) + f_x(a,b,c)(x-a) + f_y(a,b,c)(y-b) + f_z(a,b,c)(z-c) \end{aligned}$$

Case 3: Vector-valued function of multiple variables $\boldsymbol{f} : \boldsymbol{R}^n \to \boldsymbol{R}^m$

A vector-valued function returns a vector as its output. The notation $\boldsymbol{f} : \boldsymbol{R}^n \to \boldsymbol{R}^m$ indicates that the function f takes a n-dimensional vector as input and produces a m-dimensional vector $\boldsymbol{f}(\boldsymbol{x})$ as output. Ex. $f(x, y, z) = (x-y, xz), f(1, 2, 3) = (-1, 3)$

The vector-valued function $\boldsymbol{f}(\boldsymbol{x})$, whose output is a vector of m components, can be written as

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}) &= \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ f_3(\boldsymbol{x}) \\ \vdots \\ f_m(\boldsymbol{x}) \end{bmatrix} \\ &= \begin{bmatrix} f_1(x_1, x_2, x_3, \cdots, x_n) \\ f_2(x_1, x_2, x_3, \cdots, x_n) \\ f_3(x_1, x_2, x_3, \cdots, x_n) \\ \vdots \\ f_m(x_1, x_2, x_3, \cdots, x_n) \end{bmatrix} \end{aligned}$$

The linear approximation for the vector-valued function $\boldsymbol{f}(\boldsymbol{x})$ at the operating point $\boldsymbol{a}$ is:

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}(\boldsymbol{a}) + J_{\boldsymbol{f}}(\boldsymbol{a})(\boldsymbol{x}-\boldsymbol{a}) \tag{2.4}$$

Where:

$\boldsymbol{f(x)}$	is a $m \times 1$ column vector
$\boldsymbol{f(a)}$	is a $m \times 1$ column vector of size with each component evaluated at the operating point
$J_{\boldsymbol{f}}(\boldsymbol{a})$	is a $m \times n$ Jacobian matrix - also called the matrix of partial derivatives
$(\boldsymbol{x} - \boldsymbol{a})$	is the first order term of the taylor series expansion - a n $\times$1 column vector
$\boldsymbol{x}$	is a $n \times 1$ column vector (variables of f)
$\boldsymbol{a}$	is a $n \times 1$ column vector (operating points)

Each component of the vector-valued function $\boldsymbol{f(x)}$ is a scalar-valued function. The matrix of partial derivatives of each component $f_i(\boldsymbol{x})$ would be a $1 \times n$ row matrix. Stacking these row matrices on top of each other to form a larger matrix, the full $m \times n$ Jacobian matrix at the operating points $\boldsymbol{x}$=$\boldsymbol{a}$ is

$$J_{\boldsymbol{f}}(\boldsymbol{a}) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(\boldsymbol{a}) & \frac{\partial f_1}{\partial x_2}(\boldsymbol{a}) & \cdots & \frac{\partial f_1}{\partial x_n}(\boldsymbol{a}) \\ \frac{\partial f_2}{\partial x_1}(\boldsymbol{a}) & \frac{\partial f_2}{\partial x_2}(\boldsymbol{a}) & \cdots & \frac{\partial f_2}{\partial x_n}(\boldsymbol{a}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\boldsymbol{a}) & \frac{\partial f_m}{\partial x_2}(\boldsymbol{a}) & \cdots & \frac{\partial f_m}{\partial x_n}(\boldsymbol{a}) \end{bmatrix} \tag{2.5}$$

2.2 Discretizing via Numerical Methods

2.2.1 Euler's Method

In real applications we would not use a simple method such as Euler's. The simplest method that would probably be used in a real application is the standard Runge–Kutta method. That is a fourth order method, meaning that if we halve the interval, the error generally goes down by a factor of 16.

Reference: `https://web.uvic.ca/~tbazett/diffyqs/numer_section.html`

2.2.2 Runge-Kutta Method