Kalman Filter Introduction

Note: This section is currently under revision.

While the Kalman Filter is simple and elegant, it is not always obvious what it is doing conceptually or mathematically. Constructing your own with a guide, or using a code package with instructions is quite possible to do without understanding the filter. However, when choices must be made about the code, the hardware, or the values, or when general problems arise, a more thorough understanding becomes paramount.

The true algorithm for the Kalman filter is covered in the Kalman Filter section. This introduction will instead incrementally construct an equivalent algorithm starting from the concept of simple Linear Least Square Estimation, using only basic matrix operations and basic statistics.

Section 1.1 - Linear Least Squares Estimation

The Kalman Filter relies on a simple underlying concept – the linear least squares estimation. Given multiple noisy measurements of some state (speed, depth, acceleration, voltage, etc) the **LLSE** is an estimate that optimizes for the minimum of the sum of the squares of the errors.

In more formal terms, for some m measurements Y that are linear functions of a system with n unknown states X where $m>=n$. Such systems are said to be *over-determined,* whereby it is impossible to choose values of X that will satisfy every measurement perfectly, and thus a compromise of values of X is chosen that minimizes the total sum of the squares of the error between each measurement

$$ Y_{est} = \text{arg}\,\min\limits_{\beta}\sum\|y-X\beta||^2 $$

Given the matrix format:

$$ \beta X = Y $$

$$ \begin{equation} \label{eq:control:expanded} \begin{bmatrix} \beta_{1,1} & \beta_{1,2} & \dots & \beta_{1,n} \\ \beta_{2,1} & \beta_{2,2} & \dots & \beta_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{m,1} & \beta_{m,2} & \dots & \beta_{m,n} \end{bmatrix} \begin{bmatrix} X_{1} \\ X_{2} \\ \vdots \\ X_{n} \end{bmatrix} = \begin{bmatrix} Y_{1} \\ Y_{2} \\ \vdots \\ Y_{m} \\ \end{bmatrix} \end{equation} $$ This calculation could be performed iteratively, and the minimizing X_{est} discovered, but with the measurements enumerated in this format, matrix arithmetic offers us a simple way to solve for X_{est}. For those that recall their Geometry class in high school, to solve for X we need simply invert β and multiply that inverse by both sides.

$$ \beta^{-1} \beta X = \beta^{-1} Y \\ IX = \beta^{-1} Y \\ X = \beta^{-1} Y $$

However, you can only invert square matrices. For all $m \neq n$ this won't be the case. Here we employ the Moore-Penrose LLSE calculation.

First both sides are multiplied by the transpose of β.

$$\beta' \beta X = \beta' Y $$

Recall: $$ \beta' = \begin{bmatrix} \beta_{1,1} & \beta_{2,1} & \dots & \beta_{m,1} \\ \beta_{1,2} & \beta_{2,2} & \dots & \beta_{n,1} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{1,n} & \beta_{2,n} & \dots & \beta_{n,m} \end{bmatrix} $$

$\beta' \beta$ will be a square matrix of size n. Assuming that at least one measurement of all states X have been included, and indicated in β this new square matrix will be invertable. We can then multiply both sides by that inverse to isolate the X state vector.

$$ (\beta' \beta)^{-1}\beta' \beta X = (\beta' \beta)^{-1}\beta' Y \\ X = (\beta' \beta)^{-1}\beta' Y $$

Remarkably, this equation will give us an X vector that satisfies the LLSE optimization.

Let's try an example:

If we want to fit a line to two points, both data points must satisfy the line equation $y = mx + b$.

$$y_1 = mx_1 + b, \\ y_2 = mx_2+b $$

We can pose the mathematical question in a matrix-format:

$$ \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \end{bmatrix} \begin{bmatrix} m \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} $$

Here our unknown slope and y-intercept m and b form our unknown state vector X. Our dependent measurements y are used to construct our measurement vector Y, and the linear combination of independent variables and constants that relate that measurement to our state form β.

Given the two coordinate pairs $(-2, -\frac{8}{3})$ and $(4, -\frac{2}{3})$ we can find the line that is defined by them.

$$ \beta = \begin{bmatrix} -2 & 1 \\ 4 & 1 \end{bmatrix},\qquad X = \begin{bmatrix} m \\ b \end{bmatrix},\qquad Y = \begin{bmatrix} -\frac{8}{3} \\ -\frac{2}{3} \end{bmatrix} \\ \beta^{-1}Y = X \\ \begin{bmatrix} -2 & 1 \\ 4 & 1 \end{bmatrix}^{-1} \begin{bmatrix} -\frac{8}{3} \\ -\frac{2}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} \\ -2 \end{bmatrix} $$

Thus our slope $m = \frac{1}{3}$ and our linear offset $b = -2$.

From:
https://robosub.eecs.wsu.edu/wiki/ - **Palouse RoboSub Technical Documentation**

Permanent link:
https://robosub.eecs.wsu.edu/wiki/cs/localization/kalman/introduction/start?rev=1484610333

Last update: **2017/01/16 15:45**