Kalman Filter Algorithm

Note: This section is currently under revision.

This section covers the Kalman Filter Algorithm. First we'll cover the State Space format of modeling and measuring a discrete-time dynamic system of estimated states, noisy inputs, and noisy measurements. Second, we'll explore all the different pieces of information about our system necessary to inform the algorithm. Third, the specific Kalman Filter Algorithm constructed based off of those parameters. Finally, we'll use some example state spaces and measurements to see how well we track.

Note: all images below have been created with simple Matlab Scripts. If seeing the code helps clarify what's going on, the .m files can all be found under internal location cs:localization:kalman.

Section 1 - State Space Format

The State Space Format is a universally standardized format for dynamic systems in the signals and controls community. In the continuous-time domain, the derivative of the state $\dot{x}(t)$ is a linear function of $x(t)$. In the discrete-time domain, where we'll be operating, the next state $x[k+1]$ is a linear function of the current state $x[k]$.

$$ X_{k+1} = AX_k + B(u_k + \sim\mathcal{N}(0,Q^2)) \\ Y_k = CX_k + \sim\mathcal{N}(0,\sigma^2) $$

Vector X_k is the State Vector which contains all states of the system at time-step k. These include things like position, velocity, orientation, voltage, etc. Matrix A is the Transmission Matrix, which contains the dynamics of the system, and calculates the next state given the current state. B is the input matrix, which describes the dynamics of inputs u.

Vector Q is the Process Noise of the system, which is the combination of the variance of the inputs σ_u^2 and an estimated degree of possible external forces σ_ext^2. Random forces from bumping into walls, random currents in the water, diver interaction, and other unpredictable perturbations. Put another way, Q describes the uncertainty involved when predicting into to future, even given perfect information about the present State. Were there no external forces, perfect actuators, and a perfect initial state x_0, the system state could be predicted perfectly into the future. This is obviously not the case in the real world, hence the need to specify uncertainty in the model, and thus in predicting the future states.

Vector Y_k is the measurement vector. It contains the values taken from the sensors. Matrix C is the Emission Matrix.

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

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

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.

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

$\beta' \beta$ 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.