Tutorial 5: Linear Quadratic Estimator (LQE)

This tutorial will go over the key concept of the Linear Quadratic Estimator (LQE), also called Kalman filter. It will show how this powerful technique can be used to model sensory prediction and sensor fusion in the human central nervous system (CNS). Two MATLAB scripts (*Tutorial_5_students.m* & *KalmanFilter.m*) have been uploaded on BlackBoard that can be used as a starting point for your simulations.

The deadline to upload your solution of this tutorial to blackboard is **March 8th 2018 at 23h59m**.

Background

Although many types of filters exist, LQE holds a special spot for engineers. As an algorithm, LQE is nothing but a least square estimator that will minimise the difference between an observed variable and an estimation of that observed variable. However, what makes LQE special compared with other filters is that it is using the statistical information of the signal that is being fed to it. For example, an exponentially weighted moving average filter will simply weight the most recent observations more heavily than those before but will not consider the statistics of the noisy signal it is filtering. On the other hand LQE has a model of the available signals statistics and it takes advantage of this to get better estimates of the "true", i.e. the ideal noiseless signal. It does this by predicting what the next observation will be and optimally correcting itself by comparing its estimate with the observation of the noisy signal. Figure 1 shows how the LQE removes the signal noise and enables it to follow the unknown 'true' signal.

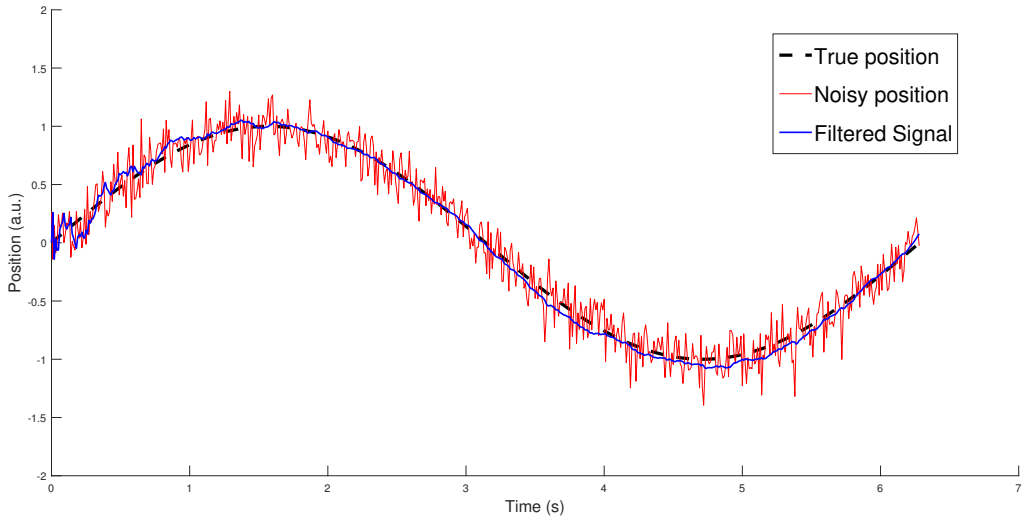

Figure 1: **An imaginary camera's measurements of a moving butterfly corrupted with zero mean Gaussian noise. One can observe that the LQE requires some time to converge to the optimal value.**

LQE is described through *i*) a *state equation*

$$\mathbf{z}_{j+1} = \mathbf{A}\,\mathbf{z}_j + \boldsymbol{\omega}_j \ , \quad \boldsymbol{\omega} \in \mathcal{G}(\mathbf{0}, \mathbf{Q}) \tag{1}$$

where $j \in \mathbb{N}$ is the time index, $\mathbf{z}$ the *state vector*, $\mathbf{A}$ the *transition matrix*, $\boldsymbol{\omega}$ a vector of (unbiased Gauss distributed) *system noise* with associated covariance matrix $\mathbf{Q} \equiv E[\boldsymbol{\omega}\,\boldsymbol{\omega}^T]$, and *ii*) an *observation equation*

$$\mathbf{y}_j = \mathbf{C}\,\mathbf{z}_j + \boldsymbol{\nu}_j \ , \quad \boldsymbol{\nu} \in \mathcal{G}(\mathbf{0}, \mathbf{R}) \tag{2}$$

where $\mathbf{y}$ is the *observation*, $\mathbf{C}$ a projection matrix and $\boldsymbol{\nu}$ a vector of (unbiased Gauss distributed) *sensor noise* with covariance matrix $\mathbf{R} \equiv E[\boldsymbol{\nu}\,\boldsymbol{\nu}^T]$. These equations are abstract and may thus be difficult to understand at first, so we start by introducing a simple system and examine how LQE can deal with it.

Imagine the following situation: you have set up a camera in the garden which is capturing live footage of a rare butterfly. Because of your life-long obsession with butterflies you feel compelled to follow its exact flight path. How can you accomplish such a feat? Fig.1 illustrates how difficult it may appear to retrieve the 'true' value of a noisy signal. First, we formalise the measurement process through the observation equation, which describes what information the LQE will receive as a measurement to infer the true signal. From the camera we can obtain the noisy position of the butterfly that will feed into the filter to measure the position:

$$y_j = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{z}_j + \nu_j\,, \quad \nu_j \in \mathcal{G}(0, \sigma_\nu^2) \tag{3}$$

where the observation y has only one row in this example and ν has covariance $E[\nu\,\nu] = \sigma_\nu^2$. To visualise noise ν you can think of it as random numbers corrupting the ideal 'true signal'.

To best use this noisy measurement we need to model the butterfly movement. For simplicity, let us start with a basic model of this movement in 1 dimension. Assuming the butterfly flies a smooth trajectory without sudden jerks, it can be described by another stochastic equation:

$$\begin{bmatrix} p \\ v \\ a \end{bmatrix}_{j+1} = \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p \\ v \\ a \end{bmatrix}_j + \boldsymbol{\omega}_j\,, \quad \boldsymbol{\omega}_j \in \mathcal{G}(\mathbf{0}, \mathbf{Q}) \tag{4}$$

where $\mathbf{z} \equiv [p, v, a]^T$ is the state vector describing the butterfly's position, velocity and acceleration, and $\boldsymbol{\omega}_j$ is system noise. This equation describes how the state evolves over time. The process noise enables the system to evolve by adapting the parameters of the second order motion tracker defined by above equation.

Using the linear framework of the state and observation equations eqs.(1, 2), we now define LQE as an algorithm to best estimate $\hat{\mathbf{z}}_j$ by minimising the square estimation error $E[||\mathbf{z}_j - \hat{\mathbf{z}}_j||^2]$, using the following iterative equation:

$$\begin{aligned} \mathbf{K}_{j+1} &= \mathbf{P}_j \mathbf{C}^T \left(\mathbf{C}\mathbf{P}_j\mathbf{C}^T + \mathbf{R}\right)^{-1} \\ \hat{\mathbf{z}}_{j+1} &= \mathbf{A}\,\hat{\mathbf{z}}_j + \mathbf{K}_{j+1}(\mathbf{y}_j - \mathbf{C}\,\mathbf{A}\,\hat{\mathbf{z}}_j) \\ \mathbf{P}_{j+1} &= \mathbf{A}\left(\mathbf{P}_j - \mathbf{P}_j\mathbf{C}^T(\mathbf{C}\mathbf{P}_j\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}\mathbf{P}_j\right)\mathbf{A}^T + \mathbf{Q} \end{aligned} \tag{5}$$

where $\mathbf{K}$ is the Kalman gain matrix and other variables are defined in eqs.(1, 2).

LQE is commonly used for motion prediction and for sensor fusion where multiple measurements of the same state are combined optimally. It has been developed independently by several people in the 1960s for better estimating the movement of rockets. It is generally applied to efficiently combine information from multiple sensors, for example the readings from a GPS sensor together with data from another sensor measuring the movement of your car's wheels, and fuse them together to get a better estimate for your position compared with either sensor alone.

Questions

We will now develop an LQE model of sensory processing in humans. A program with a moving object and a LQE to test this model is provided on blackboard. Values for variables not indicated in the text are defined in the Matlab script.

(a) We first examine how the CNS could utilise a noisy signal in order to detect a smoother movement. For instance, humans may use vision to follow the butterfly trajectory for which we take here a simpler sinusoidal wave. To visualise this situation run the Tutorial 5 Matlab code and add the observation from a signal with noise v and $\sigma_v = 1.5$ cm into the LQE. Plot the true butterfly's trajectory (unknown to the observator), the noisy sensory information received and the filtered signal as functions of time. Then, repeat the same procedure for $\sigma_v = 4.5$ cm, plot the noisy and filtered signals as functions of time in the same figure. The results should show how the signal is tracked using only visual feedback. Describe how the convergence of the filtered signal depends on the initial covariance matrix, by trying values such as $P_0 = diag\,(10^5, 10^5, 10^5)$ and $P_0 = diag(1, 1, 1)$.

[25 marks]

(b) Now we consider that the CNS uses both vision and proprioception to track the butterfly movement, i.e. that it can combine the information from these two sensory modalities. We first assume that vision and proprioception provide independent measurements, thus

$$\mathbf{R} = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_p^2 \end{bmatrix}, \tag{6}$$

and are subjected to the same level of noise with $\sigma_v = \sigma_p = 1.5$ cm. To compare the filtering with only vision or with additional proprioception, plot the estimation of the LQE in these two cases, and compare the mean square error relative to the ideal movement. *Hint*: As the system has two observations, you will need to modify the projection matrix in the observation equation correspondingly.

[30 marks]

(c) One can expect that the level of noise related to proprioception is larger, so we set $\sigma_p = 4.5$ cm. Plot in the same figure the following time series: the clean signal, the filtered signal with only visual feedback, and the filtered signal with vision and noisy proprioception. Does proprioception improve the estimation of the true signal in this case? Justify your answer by computing the mean square error relative to the ideal movement.

[15 marks]

(d) We live in the past, in the sense that we perceive and act based on delayed sensory information. As we are nevertheless able to carry out successful actions in real-time, e.g. to catch the butterfly, this means that CNS compensates for the sensory delays. We can interpret this fact in our LQE modelling, by assuming that the CNS compensates for the delay as follows:

$$\mathbf{z}_j^* = \mathbf{A}^\delta \, \hat{\mathbf{z}}_{j-\delta} \tag{7}$$

where $\mathbf{z}_j^*$ is the prediction of the state at time j, and the LQE is computed iteratively with delayed sensory information using

$$\hat{\mathbf{z}}_{j+1-\delta} = \hat{\mathbf{z}}_{j-\delta} + \mathbf{K}_{j+1} \left(\mathbf{y}_{j-\delta} - \mathbf{C} \, \hat{\mathbf{z}}_{j-\delta} \right). \tag{8}$$

We propose to test the effect of a sensory delay $\delta = 0.1$ s and its compensation on the visual feedback alone. In this purpose, plot the true signal, the delayed and noisy visual signal, and the estimate obtained by implementing the above mentioned prediction algorithm. *Hint*: As the sensory signal is delayed of a time δ, the predicted state will be available only from that time.

[30 marks]