Kalman Filter for estimating tank outflow

Finn Haugen
finn@techteach.no

22. March 2008

1 The Kalman Filter algorithm

Predicted measurement estimate:

$$y_p(k) = g\left[x_p(k)\right] \tag{1}$$

Innovation variable:

$$e(k) = y(k) - y_p(k) \tag{2}$$

Corrected state estimate (used as estimate in applications):

$$x_c(k) = x_p(k) + Ke(k) \tag{3}$$

Predicted state estimate:

$$x_p(k+1) = f[x_c(k),\, u(k)] \tag{4}$$

It will be assumed that K is the steady-state Kalman Filter gain. The information needed to compute the steady-state Kalman Filter gain is shown in Figure 1. It can be calculated with e.g. the **Kalman Gain** function in LabVIEW Control Design Toolkit, or with the **kalman** function in LabVIEW MathScript or with the **dlqe** function in Matlab.

2 Application of Kalman Filter: Estimation of outflow

Figure 2 shows a liquid tank.

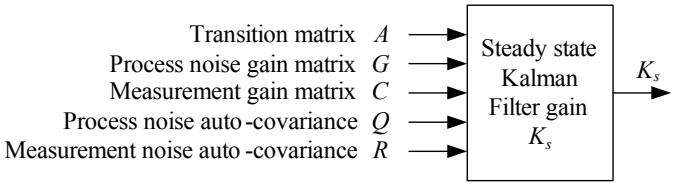

Figure 1: Illustration of what information is needed to compute the steady-state Kalman Filter gain, K_s.

We will design a steady state Kalman Filter to estimate the outflow F_{out}. The level h is measured.

Mass balance of the liquid in the tank is (mass is ρAh)

$$\rho A_{tank}\dot{h}(t) = \rho K_p u - \rho F_{out}(t) \tag{5}$$
$$= \rho K_p u - \rho F_{out}(t) \tag{6}$$

After cancelling the density ρ the model is

$$\dot{h}(t) = \frac{1}{A_{tank}}\left[K_p u - F_{out}(t)\right] \tag{7}$$

We assume that the unknown outflow is slowly changing, almost constant. We define the following augmentative model:

$$\dot{F}_{out}(t) = 0 \tag{8}$$

The model of the system is given by (7) – (8). Although it is not necessary, it is convenient to rename the state variables using standard names. So we define

$$x_1 = h \tag{9}$$

$$x_2 = F_{out} \tag{10}$$

The model (7) – (8) is now

$$\dot{x}_1(t) = \frac{1}{A_{tank}}\left[K_p u(t) - x_2(t)\right] \tag{11}$$

$$\dot{x}_2(t) = 0 \tag{12}$$

Applying Euler Forward discretization with time step T and including white disturbance noise in the resulting difference equations yields

$$x_1(k+1) = \underbrace{x_1(k) + \frac{T}{A_{tank}}\left[K_p u(k) - x_2(k)\right]}_{f_1(\cdot)} + w_1(k) \tag{13}$$

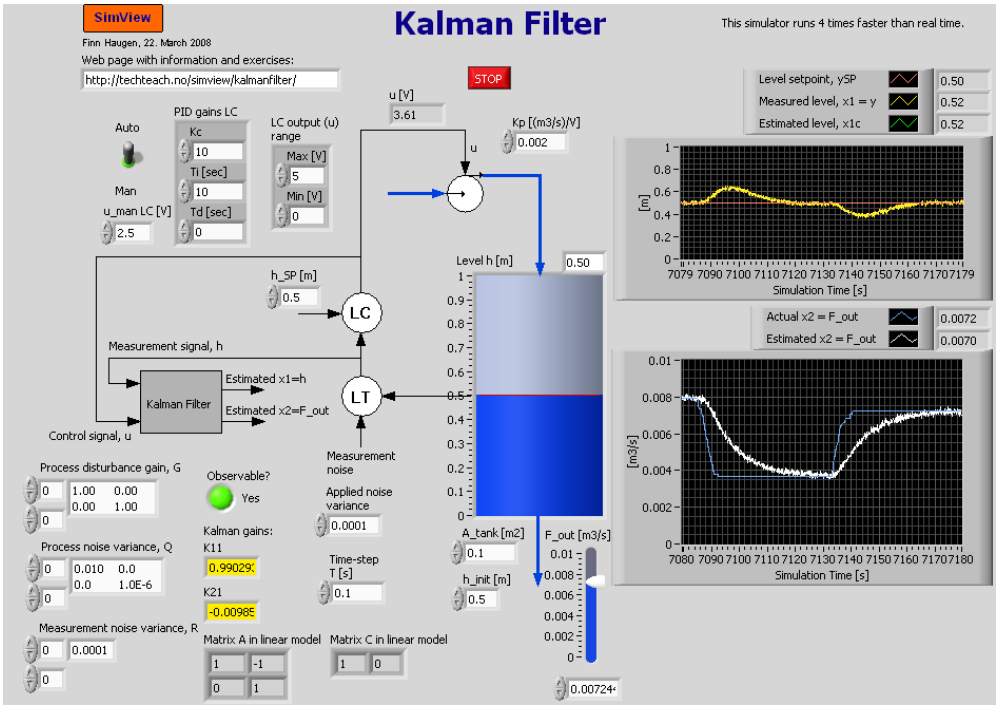

Figure 2: Liquid tank with Kalman Filter (and with level control system)

$$x_2(k+1) = \underbrace{x_2(k)}_{f_2(\cdot)} + w_2(k) \tag{14}$$

or

$$x(k+1) = f\left[x(k), u(k)\right] + w(k) \tag{15}$$

w_1 and w_2 are independent (uncorrelated) white process noises with assumed variances $R_{w_1}(L) = Q_1\delta(L)$ and $R_{w_2}(L) = Q_2\delta(L)$ respectively. Here, Q_1 and Q_2 are variances. The multivariable noise model is then

$$w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \tag{16}$$

with auto-covariance

$$R_w(L) = Q\delta(L) \tag{17}$$

where

$$Q = \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{22} \end{bmatrix} \tag{18}$$

Assuming that the level x_1 is measured, we add the following measurement equation to the state space model:

$$y(k) = g\left[x_p(k), u(k)\right] + v(k) = x_1(k) + v(k) \tag{19}$$

where v is white measurement noise with assumed variance

$$R_v(L) = R\delta(L) \tag{20}$$

where R is the measurement variance.

The following numerical values are used:

$$\text{Sampling time: } T = 0.1 \text{ s} \tag{21}$$

$$A_{tank} = 0.1 \text{ m}^2 \tag{22}$$

$$K_p = 0.001 \text{ (m}^3\text{/s)/V} \tag{23}$$

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \text{ (initally, may be adjusted)} \tag{24}$$

$$R = 0.0001 \text{ m}^2 \text{ (Gaussian white noise)} \tag{25}$$

We will set the initial estimates as follows:

$$x_{1_p}(0) = x_1(0) = y(0) \text{ (from the sensor)} \tag{26}$$

$$x_{2_p}(0) = 0 \text{ (assuming no information about initial value} \tag{27}$$

The Kalman Filter algorithm is as follows: The predicted level measurement is calculated according to (1):

$$y_p(k) = g\left[x_p(k), u(k)\right] = x_{1_p}(k) \tag{28}$$

with initial value as given by (26). The innovation variable is calculated according to (2), where y is the level measurement:

$$e(k) = y(k) - y_p(k) \tag{29}$$

The corrected state estimate is calculated according to (3):

$$x_c(k) = x_p(k) + Ke(k) \tag{30}$$

or, in detail:

$$\begin{bmatrix} x_{1_c}(k) \\ x_{2_c}(k) \end{bmatrix} = \begin{bmatrix} x_{1_p}(k) \\ x_{2_p}(k) \end{bmatrix} + \underbrace{\begin{bmatrix} K_{11} \\ K_{21} \end{bmatrix}}_{K} e(k) \tag{31}$$

This is the applied esimate!

The predicted state estimate for the next time step, $x_p(k+1)$, is calculated according to (4):

$$x_p(k+1) = f[x_c(k),\, u(k)] \tag{32}$$

or, in detail:

$$\begin{bmatrix} x_{1_p}(k+1) \\ x_{2_p}(k+1) \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_{1_c}(k) + \frac{T}{A_{tank}} \left[K_p u(k) - x_{2_c}(k) \right] \\ x_{2_c}(k) \end{bmatrix} \tag{33}$$

To calculate the steady state Kalman Filter gain K_s the following information is needed, cf. Figure 1:

$$A = \left. \frac{\partial f(\cdot)}{\partial x} \right|_{x_p(k),\, u(k)} \tag{34}$$

$$= \left. \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \right|_{x_p(k),\, u(k)} \tag{35}$$

$$= \begin{bmatrix} 1 & -\frac{T}{A_{tank}} \\ 0 & 1 \end{bmatrix} \tag{36}$$

$$G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2 \text{ (identity matrix)} \tag{37}$$

$$C = \left. \frac{\partial g(\cdot)}{\partial x} \right|_{x_p(k),\, u(k)} \tag{38}$$

$$= \begin{bmatrix} 1 & 0 \end{bmatrix} \tag{39}$$

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix} \text{ (initially, may be adjusted)} \tag{40}$$

$$R = 0.001 \text{ m}^2 \tag{41}$$

Figure 2 shows the front panel of a LabVIEW simulator of this example. The outflow $F_{out} = x_2$ was changed during the simulation, and the Kalman Filter estimates the correct steady state value (the estimate is however noisy). During the simulations, I found that (40) gave too noise estimate of $x_2 = F_{out}$. I ended up with

$$Q = \begin{bmatrix} 0.01 & 0 \\ 0 & 10^{-6} \end{bmatrix} \tag{42}$$

as a proper value.

Figure 3 shows how the steady state Kalman Filter gain K_s is calculated using the **Kalman Gain** function. The figure also shows how to check for observability with the **Observability Matrix** function. Figure 4 shows the implementation of the Kalman Filter equations in a Formula Node in LabVIEW.

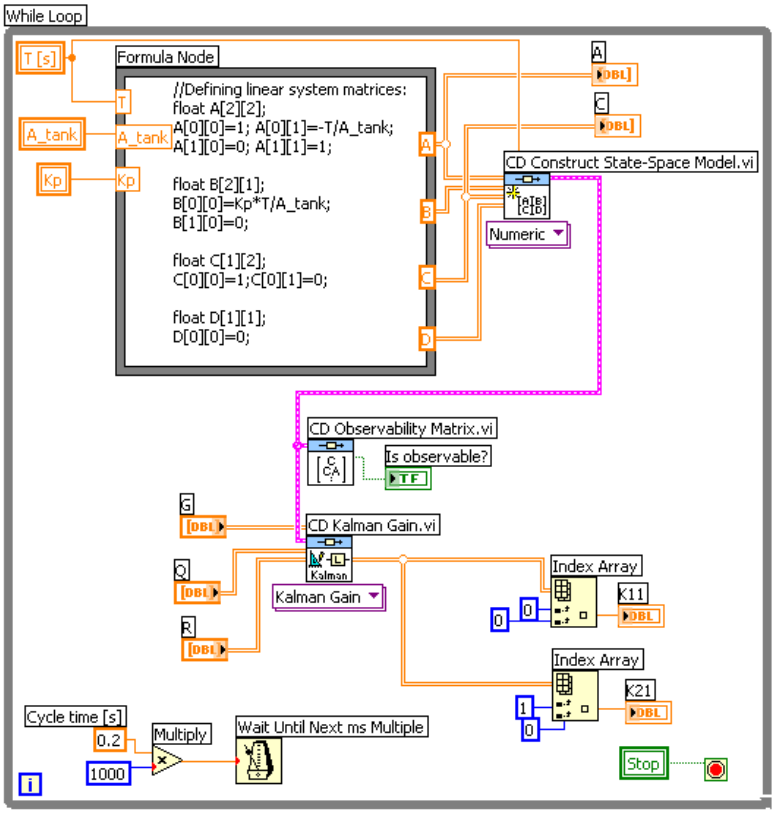

Figure 3: Calculation of the steady state Kalman Filter gain with the Kalman Gain function, and checking for observability with the Observability Matrix function.

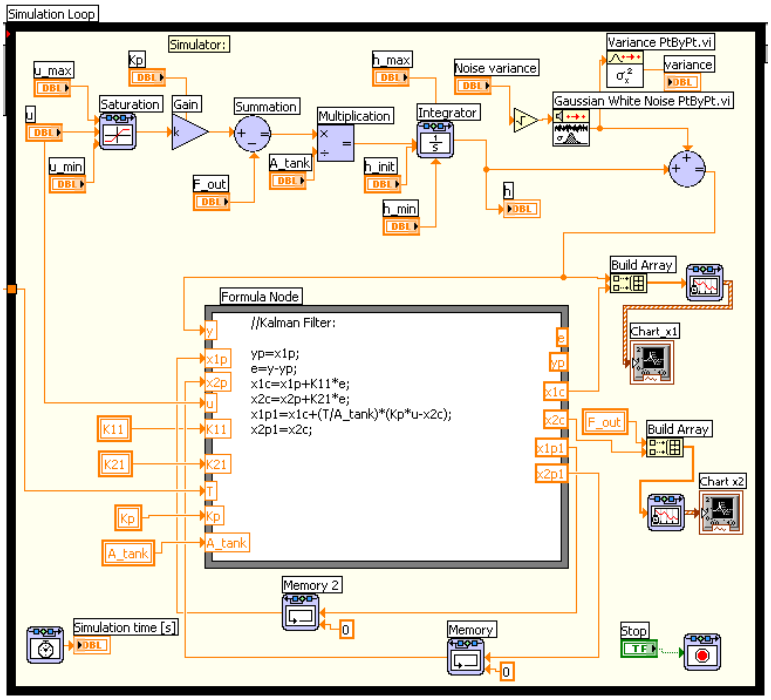

Figure 4: Implementation of the Kalman Filter equations in a Formula Node. (The Kalman Filter gain is fetched from the While loop in the Block diagram using local variables.)