Numerical Example - LPV Kalman Filter

Octavio Adrián Hernández Gómez - adrianhernandezg@hotmail.com

1 Nonlinear System

Consider the following Nonlinear System in continous time perturbed by some zero-mean Gaussian process and measurement noise $w(t)$ and $v(t)$. Our goal is to estimate the system states and attenuate the impact of such noises.

$$\dot{x}_1(t) = -x_1(t) + x_2^2(t) + w(t) \tag{1}$$
$$\dot{x}_2(t) = -x_2(t) + u(t) + w(t) \tag{2}$$
$$y(t) = x_1(t) + v(t) \tag{3}$$

Taking into account sampled measurements for digital implementation, a discrete-time representation is needed. Here, the Heun's discretization method is considered

$$x_1(k+1) = x_1(k) + (\frac{\Delta t}{2})\Big[\Big(-x_1(k) + x_2^2(k)\Big) + \Big(-x_{1E}(k) + x_{2E}^2(k)\Big)\Big] \tag{4}$$
$$x_2(k+1) = x_2(k) + (\frac{\Delta t}{2})\Big[\Big(-x_2(k) + u(k)\Big) + \Big(-x_{2E}(k) + u(k)\Big)\Big] \tag{5}$$

where:

$$x_{1E}(k) = x_1(k) + \Delta t\Big(-x_1(k) + x_2^2(k)\Big) \tag{6}$$
$$x_{2E}(k) = x_2(k) + \Delta t\Big(-x_2(k) + u(k)\Big) \tag{7}$$

The next step is to obtain an LPV representation, which will be used later for the design of the LPV Kalman filter. Consider the following one:

$$x(k+1) = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \frac{\Delta t}{2}\left(\begin{bmatrix} -1 & x_2(k) \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} -1 & x_{2E}(k) \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_{1E}(k) \\ x_{2E}(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k)\right) \tag{8}$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix}\begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{9}$$

Here, the only scheduling variable is $x_2(k)$, so that $l = 1$. Thereby, the binary configuration is equivalent to a 1-bit truth table, delivering into 2 vertices.

Table 1: Vertex configuration

Vertex	**Binary**	**Vertex System**	**Gain Scheduling Function**
1	0	x_{2min}	$\psi_1(\varphi(k)) = (\frac{x_{2max} - x_2}{x_{2max} - x_{2min}})$
2	1	x_{2max}	$\psi_2(\varphi(k)) = (1 - \psi_1(\varphi(k)))$

Let's consider that x_2 may vary over $\in [0.1, 0.3]$ under a given input $u(k) = 0.1sin(0.16k) + 0.2$. The *vertex systems* are constructed so as to exactly reproduce the original nonlinear system, yielding an equivalent polytopic representation.

$$\begin{cases} x(k+1) = \sum_{i=1}^{2^l} \psi_i(\varphi(k)) \left[A_{d_i} x(k) + B_{d_i} u(k)\right] + w_d(k) \\ y(k) = \sum_{i=1}^{2^l} \psi_i(\varphi(k)) \left[C_{d_i} x(k)\right] + v_d(k). \end{cases} \tag{10}$$

In this case, in order to calculate the *vertex systems* supose that $x_1 = x_{1E}$ and $x_2 = x_{2E}$.

For $x_2 = x_{2min} = 0.1$, with $\Delta t = 0.01$

$$x(k+1) = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \frac{\Delta t}{2} \left(\begin{bmatrix} -1 & 0.1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} -1 & 0.1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \right) \tag{11}$$

$$x(k+1) = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \frac{\Delta t}{2} \left(\begin{bmatrix} -2 & 0.2 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u(k) \right) \tag{12}$$

$$x(k+1) = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{\Delta t}{2} \begin{bmatrix} -2 & 0.2 \\ 0 & -2 \end{bmatrix} \right) \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.0100 \end{bmatrix} u(k) \tag{13}$$

$$x(k+1) = \begin{bmatrix} 0.9900 & 0.0010 \\ 0 & 0.9900 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.0100 \end{bmatrix} u(k) \tag{14}$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{15}$$

For $x_2 = x_{2max} = 0.3$, with $\Delta t = 0.01$

$$x(k+1) = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \frac{\Delta t}{2}\left(\begin{bmatrix} -1 & 0.3 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) + \begin{bmatrix} -1 & 0.3 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(k) \right) \tag{16}$$

$$x(k+1) = \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \frac{\Delta t}{2}\left(\begin{bmatrix} -2 & 0.6 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 2 \end{bmatrix} u(k) \right) \tag{17}$$

$$x(k+1) = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{\Delta t}{2} \begin{bmatrix} -2 & 0.6 \\ 0 & -2 \end{bmatrix} \right) \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.0100 \end{bmatrix} u(k) \tag{18}$$

$$x(k+1) = \begin{bmatrix} 0.9900 & 0.0030 \\ 0 & 0.9900 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \begin{bmatrix} 0 \\ 0.0100 \end{bmatrix} u(k) \tag{19}$$

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} \tag{20}$$

They can also be computed using the comand *c2d* in *MATLAB*. Nevertheless, in (10) it is assumed that the gain scheduling functions depend on available online signals included in the scheduling vector. However, the focus here is on the case where the scheduling vector is not fully measurable and its variables are required. The aforementioned system can be expressed in terms of scheduling functions that depend on the estimated scheduling vector:

$$\begin{cases} x(k+1) = \sum_{i=1}^{2^l} \psi_i(\hat{\varphi}(k)) \left[A_{d_i} x(k) + B_{d_i} u(k) \right] + \Delta w(k) \\ y(k) = \sum_{i=1}^{2^l} \psi_i(\hat{\varphi}(k)) \left[C_{d_i} x(k) \right] + \Delta v(k) \end{cases} \tag{21}$$

by adding and subtracting:

$$\begin{aligned} \delta_w(k) &= \sum_{i=1}^{2^l} \Big(\psi_i(\varphi(k)) - \psi_i(\hat{\varphi}(k)) \Big) \left[A_{d_i} x(k) + B_{d_i} u(k) \right] \\ \delta_v(k) &= \sum_{i=1}^{2^l} \Big(\psi_i(\varphi(k)) - \psi_i(\hat{\varphi}(k)) \Big) \left[C_{d_i} x(k) \right] \\ \Delta w(k) &= \delta_w(k) + w_d(k), \quad \Delta v(k) = \delta_v(k) + v_d(k) \end{aligned} \tag{22}$$

2 LPV Kalman Filter Design

As the new polytopic system representation is based on the estimated scheduling vector, a Kalman Filter must be designed to estimate the system states while minimizing the effect of assumed stochastic perturbations and Gaussian noises on the estimation error. The next step is to ensure observability in each of the vertex.

$$rank(O_{d_i}) = rank\left(\begin{bmatrix} C_{d_i} \\ C_{d_i}A_{d_i} \\ \vdots \\ C_{d_i}A_{d_i}^{n-1} \end{bmatrix}\right) = n,\ i = 1, \dots, 2^l. \tag{23}$$

In order to construct a filter to reduce the noise in the estimation, the following LPV Kalman Filter is considered to reconstruct the state vector affected by uncertainties:

$$\begin{cases} \hat{x}(k+1) = A_d(\varphi(k))\hat{x}(k) + B_d(\varphi(k))u(k) \\ \qquad\qquad + \mathcal{L}(\varphi(k))\big(y(k) - \hat{y}(k)\big) \\ \quad \hat{y}(k) = C_d(\varphi(k))\hat{x}(k) \end{cases} \tag{24}$$

where $\hat{x}(k)$ is the estimated state at the k-th instant, and the gain takes the following polytopic form:

$$\mathcal{L}(\varphi(k)) = \sum_{i=1}^{2^l} \psi_i(\varphi(k))\mathcal{L}_i \tag{25}$$

Nevertheless, it is not feasible to compute the gain scheduling functions required for the observer gain (25) directly, as only the output $y(t) = x_1(t)$ is available. In order to compute the gain scheduling functions, the state x_2 must be known. To address this problem, the observer is defined based on the estimated scheduling vector, along with the corresponding polytopic observer gain:

$$\begin{cases} \hat{x}(k+1) = A_d(\hat{\varphi}(k))\hat{x}(k) + B_d(\hat{\varphi}(k))u(k) \\ \qquad\qquad + \mathcal{L}(\hat{\varphi}(k))\big(y(k) - \hat{y}(k)\big) \\ \quad \hat{y}(k) = C_d(\hat{\varphi}(k))\hat{x}(k) \end{cases} \tag{26}$$

$$\mathcal{L}(\hat{\varphi}(k)) = \sum_{i=1}^{2^l} \psi_i(\hat{\varphi}(k))\mathcal{L}_i. \tag{27}$$

Table 2: Vertex configuration

Vertex	**Binary**	**Vertex System**	**Estimated Gain Scheduling Function**
1	0	x_{2min}	$\psi_1(\hat{\varphi}(k)) = (\frac{x_{2max} - \hat{x}_2}{x_{2max} - x_{2min}})$
2	1	x_{2max}	$\psi_2(\hat{\varphi}(k)) = (1 - \psi_1(\hat{\varphi}(k)))$

To compute the vertex gains, it is neccesary to solve the following set of LMIs, here, MOSEK solver was used.

$$\begin{aligned}
&\min_{Y=Y^T,\, W_i} \quad \gamma \\
&\text{s.t.} \\
&\begin{bmatrix} -Y & YA_{d_i} - W_i^T C_{d_i} & Y\mathcal{H}^T & W_i^T \\ A_{d_i}^T Y - C_{d_i}^T W_i & -Y & 0 & 0 \\ \mathcal{H}Y & 0 & -I_n & 0 \\ W_i & 0 & 0 & -\mathcal{R}^{-1} \end{bmatrix} < 0 \\
&\begin{bmatrix} \gamma I_n & I_n \\ I_n & Y \end{bmatrix} > 0 \quad \forall i = 1, \ldots, 2^l
\end{aligned} \tag{28}$$

where $\mathcal{Q} = \mathcal{Q}^T > 0$ and $\mathcal{R} = \mathcal{R}^T > 0$ are the tuning covariance matrices of process and measurement uncertainties, respectively, and $\mathcal{H} = \mathcal{Q}^{1/2}$. Moreover, if the Linear Matrix Inequalities (LMIs) (29) have a feasible solution, the offline LPV Kalman optimal filter gains at each vertex can be computed as follows: $\mathcal{L}_i = Y^{-1} W_i^T$.

$$\begin{aligned}
&\min_{Y=Y^T,\, W_1\, W_2} \quad \gamma \\
&\text{s.t.} \\
&\begin{bmatrix} -Y & YA_{d_1} - W_1^T C_{d_1} & Y\mathcal{H}^T & W_1^T \\ A_{d_1}^T Y - C_{d_1}^T W_1 & -Y & 0 & 0 \\ \mathcal{H}Y & 0 & -I_2 & 0 \\ W_1 & 0 & 0 & -\mathcal{R}^{-1} \end{bmatrix} < 0 \\
&\begin{bmatrix} -Y & YA_{d_2} - W_2^T C_{d_2} & Y\mathcal{H}^T & W_2^T \\ A_{d_2}^T Y - C_{d_2}^T W_2 & -Y & 0 & 0 \\ \mathcal{H}Y & 0 & -I_2 & 0 \\ W_2 & 0 & 0 & -\mathcal{R}^{-1} \end{bmatrix} < 0 \\
&\begin{bmatrix} \gamma I_2 & I_2 \\ I_2 & Y \end{bmatrix} > 0
\end{aligned} \tag{29}$$

3 LPV Kalman Filter Algorithm

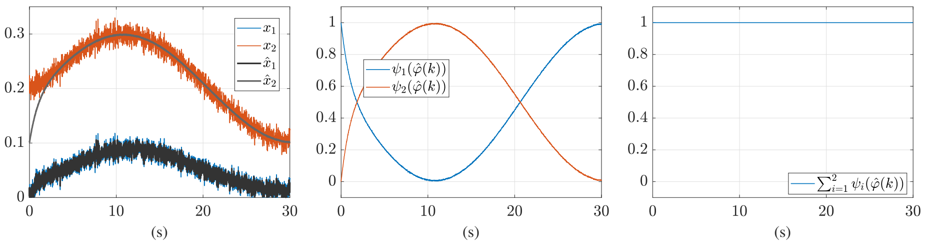
The following algorithm is based on the funcionality of the observer. A simulation with the observer code is provided in the folder

Algorithm 1 LPV Kalman Observer

```
Require: Input $u(k)$, Output $y(k)$
Ensure: State estimate $\hat{x}(k)$
 1: Compute offline optimal gains
 2: Initialize observer states $\hat{x}(0)$
 3: Define the total number of Steps
 4: Observer triggered
 5: while $k <$ Steps do
 6:     Define the bounds of the scheduling variables
 7:     Compute the estimated scheduling functions $\psi_i(\hat{\varphi}(k))$
 8:     Interpolate the observer gains
 9:     $\mathcal{L}(\hat{\varphi}(k)) \leftarrow \sum_{i=1}^{2^l} \psi_i(\hat{\varphi}(k))\, \mathcal{L}_i$
10:     Compute the LPV Kalman Filter and update the states
11:     $k \leftarrow k + 1$                                  ▷ Advance one step
12: end while
```

4 Results

Figure 1 shows the results obtained using $\mathcal{Q} = I$, $\mathcal{R} = 1$



(a) (b) (c)

Figure 1: (a) Real states and estimated; (b) Estimated gain scheduling functions; (c) Sum of the estimated gain scheduling functions

Figure 2 shows the results obtained using $\mathcal{Q} = I$, $\mathcal{R} = 100$

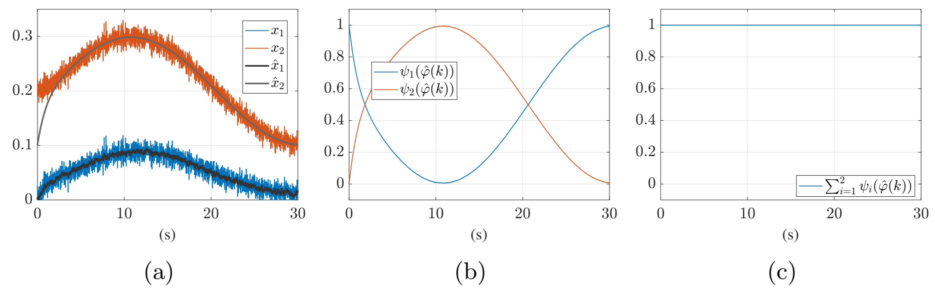


Figure 2: (a) Real states and estimated; (b) Estimated gain scheduling functions; (c) Sum of the estimated gain scheduling functions

Figure 3 shows the results obtained using $\mathcal{Q} = I$, $\mathcal{R} = 1,000$

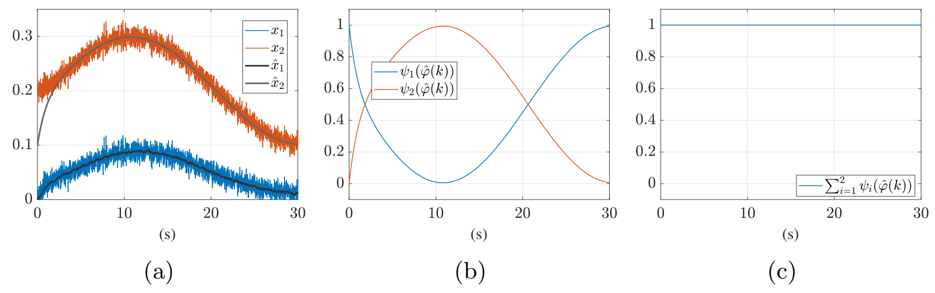


Figure 3: (a) Real states and estimated; (b) Estimated gain scheduling functions; (c) Sum of the estimated gain scheduling functions

Figure 4 shows the results obtained using $\mathcal{Q} = I$, $\mathcal{R} = 10,000$

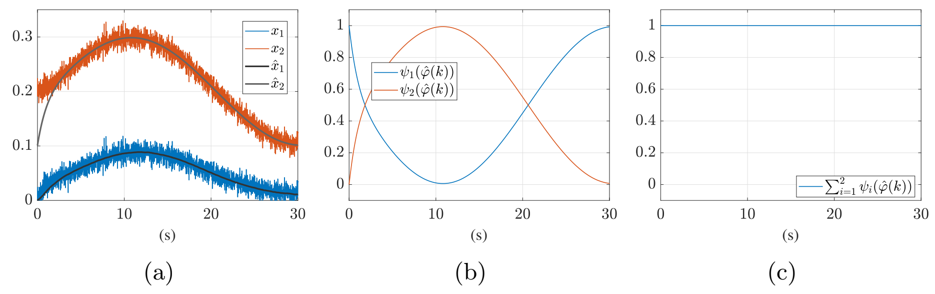


Figure 4: (a) Real states and estimated; (b) Estimated gain scheduling functions; (c) Sum of the estimated gain scheduling functions