IO realization of an LPV-SS form with static dependency

Technical report: TUE-CS-2015-001

Roland Tóth

I. Foreword

This note is about realization of a minimal *linear parameter-varying* (LPV) *state-space* (SS) form with static dependency as a *input-output* (IO) representation.

II. Realization

Consider an LPV-SS representation in the form of

$$x_{k+1} = A(p_k)x_k + B(p_k)u_k, \tag{1a}$$

$$y_k = C(p_k)x_k + Dp_k)u_k, \tag{1b}$$

where $y : \mathbb{Z} \to \mathbb{R}^{n_\mathrm{y}}$ is the output, $u : \mathbb{Z} \to \mathbb{R}^{n_\mathrm{u}}$ is the input, $x : \mathbb{Z} \to \mathbb{R}^{n_\mathrm{x}}$ is the state variable and $p : \mathbb{Z} \to \mathbb{R}^{n_\mathrm{p}}$ is the scheduling signal respectively, while $A : \mathbb{R}^{n_\mathrm{p}} \to \mathbb{R}^{n_\mathrm{x}\times n_\mathrm{x}}$, $\ldots$, $D : \mathbb{R}^{n_\mathrm{p}} \to \mathbb{R}^{n_\mathrm{y}\times n_\mathrm{u}}$ are given bounded matrix functions. As a short hand notation we will use, e.g, $A_k := A(p_k)$ to abbreviate the scheduling dependency.

The output relations can be written as follows:

$$C_{k-n_\mathrm{x}+1}x_{k-n_\mathrm{x}+1} = y_{k-n_\mathrm{x}+1} - D_{k-n_\mathrm{x}+1}u_{k-n_\mathrm{x}+1}, \tag{2a}$$

$$C_{k-n_\mathrm{x}+2}A_{k-n_\mathrm{x}+1}x_{k-n_\mathrm{x}+1} + C_{k-n_\mathrm{x}+2}B_{k-n_\mathrm{x}+1}u_{k-n_\mathrm{x}+1} = y_{k-n_\mathrm{x}+2} - D_{k-n_\mathrm{x}+2}u_{k-n_\mathrm{x}+2}, \tag{2b}$$

$$\vdots$$

$$C_k\left(\prod_{i=k-n_\mathrm{x}+1}^{k-1} A_i\right)x_{k-n_\mathrm{x}+1} + \sum_{\ell=k-n_\mathrm{x}+1}^{k-1} C_k\left(\prod_{i=\ell+1}^{k-1} A_i\right)B_\ell u_\ell = y_k - D_k u_k. \tag{2c}$$

This can be written compactly as

$$\underbrace{\begin{bmatrix} C_{k-n_\mathrm{x}+1} \\ C_{k-n_\mathrm{x}+2}A_{k-n_\mathrm{x}+1} \\ \vdots \\ C_k\left(\prod_{i=k-n_\mathrm{x}+1}^{k-1} A_i\right) \end{bmatrix}}_{\mathcal{O}_{n_\mathrm{x}}(k)} = I \underbrace{\begin{bmatrix} y_{k-n_\mathrm{x}+1} \\ \vdots \\ y_k \end{bmatrix}}_{\mathcal{Y}_{n_\mathrm{x}}(k)} -$$

$$\underbrace{\begin{bmatrix} D_{k-n_\mathrm{x}+1} & 0 & 0 & \cdots & 0 \\ C_{k-n_\mathrm{x}+2}B_{k-n_\mathrm{x}+1} & D_{k-n_\mathrm{x}+2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ C_k\left(\prod_{i=k-n_\mathrm{x}+2}^{k-1} A_i\right)B_{k-n_\mathrm{x}+1} & C_k\left(\prod_{i=k-n_\mathrm{x}+3}^{k-1} A_i\right)B_{k-n_\mathrm{x}+2} & \cdots & \cdots & D_k \end{bmatrix}}_{\mathcal{T}_{n_\mathrm{x}}(k)} \underbrace{\begin{bmatrix} u_{k-n_\mathrm{x}+1} \\ \vdots \\ u_k \end{bmatrix}}_{\mathcal{U}_{n_\mathrm{x}}(k)} \tag{3}$$

R. Tóth is with the Control Systems Group, Department of Electrical Engineering, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands, email: `r.toth@tue.nl`.

where $\mathcal{O}_{n_\mathrm{x}}(k)$ is the n_x-step observability matrix of (1). Now, under the assumption of complete observability, there exists a $\mathcal{O}^\dagger_{n_\mathrm{x}}(k)$ such that $\mathcal{O}^\dagger_{n_\mathrm{x}}(k)\mathcal{O}_{n_\mathrm{x}}(k) = I$ for all $p \in (\mathbb{R}^{n_\mathrm{p}})^\mathbb{Z}$ with left compact support and $k \in \mathbb{Z}$ defined on that support. Then, it follows that

$$x_{k-n_\mathrm{x}+1} = \mathcal{O}^\dagger_{n_\mathrm{x}}(k)\mathcal{Y}_{n_\mathrm{x}}(k) - \mathcal{O}^\dagger_{n_\mathrm{x}}(k)\mathcal{T}_{n_\mathrm{x}}(k)\mathcal{U}_{n_\mathrm{x}}(k). \tag{4}$$

By using (2a), this gives

$$y_{k-n_\mathrm{x}+1} = C_{k-\mathrm{x}+1}\mathcal{O}^\dagger_{n_\mathrm{x}}(k)\mathcal{Y}_{n_\mathrm{x}}(k) - C_{k-\mathrm{x}+1}\mathcal{O}^\dagger_{n_\mathrm{x}}(k)\mathcal{T}_{n_\mathrm{x}}(k)\mathcal{U}_{n_\mathrm{x}}(k) - D_{k-n_\mathrm{x}+1}u_{k-n_\mathrm{x}+1}. \tag{5}$$

Let us define $n_\mathrm{a} = n_\mathrm{b} = n_\mathrm{x}$ and introduce a partitioning of the above defined matrices as

$$\begin{aligned} -C_{k-\mathrm{x}+1}\mathcal{O}^\dagger_{n_\mathrm{x}}(k) &= \left[\begin{array}{ccc} \hat{A}_{n_\mathrm{a}-1}(k) & \cdots & \hat{A}_0(k) \end{array}\right], \\ -C_{k-\mathrm{x}+1}\mathcal{O}^\dagger_{n_\mathrm{x}}(k)\mathcal{T}_{n_\mathrm{x}}(k) &= \left[\begin{array}{cccc} \hat{B}_{n_\mathrm{b}}(k) + D_{k-n_\mathrm{x}+1} & \hat{B}_{n_\mathrm{b}-1}(k) & \cdots & \hat{B}_0(k) \end{array}\right], \end{aligned}$$

and define $\hat{A}_{n_\mathrm{a}} = I$. Note that the above given matrices have polynomial dynamic dependency on the backward shifted values of p. We will denote by $\diamond$ the evaluation of such a dynamic dependency w.r.t. a given trajectory of p. This gives the LPV-IO realization of (1) as

$$(\hat{A}_0 \diamond p)(k)y_k + \sum_{i=1}^{n_\mathrm{a}}(\hat{A}_i \diamond p)(k)y_{k-i} = \sum_{j=0}^{n_\mathrm{b}}(\hat{B}_j \diamond p)(k)u_{k-j}. \tag{6}$$

Note that it is often desired to have a monic representation, i.e. to guarantee that y_k is with a coefficient being the identity matrix. This can be achieved by multiplying the whole equation from the left with the inverse of $\hat{A}_0$ if it exists for all p trajectories (otherwise we can only achieve representation of the original solution set of (1) in an almost everywhere sense). The resulting form will have rational dynamic dependency in general.