A CONTROL STRATEGY USING A CPWL NOE STRUCTURE

L.R. Castro [*,1] **J. L. Figueroa** [**,1,2]
O. E. Agamennoni [***,1,3]

[*] *Dto. de Matemática, Univ. Nac. del Sur, B8000CPB Bahía Blanca,Argentina*
[**] *Dto. de Ing. Eléctrica y de Computadoras, CONICET - Univ. Nac. del Sur, B8000CPB Bahía Blanca, Argentina*
[***] *Dto. de Ing. Eléctrica y de Computadoras, CIC - Univ. Nac. del Sur, B8000CPB Bahía Blanca, Argentina*

Abstract: In this paper we develop a nonlinear control strategy based on a Nonlinear Output Error (NOE) model structure that uses Canonical High Level Piecewise Linear (HL CPWL) functions to approximate the nonlinear system. This model structure allows the implementation of identification and control algorithms that allows to increase or decrease very easily the complexity of the model during the identification process. This property is very attractive because it allows to find the appropriate NOE model without overfitting. Using this CPWL NOE model structure, we define a simple local linear control scheme.

Keywords: Nonlinear systems, identification, piecewise linear techniques.

1. INTRODUCTION

There exist a set of very well known techniques to design and analyze feedback control strategies for linear systems. If the system under consideration is nonlinear and important performance requirements are imposed, nonlinear control design tools must be used. Canonical Piecewise Linear (CPWL) approximation in the context of Nonlinear Output Error (NOE) model structure allows a systematic multilinear or Linear Parameter Varying (LPV) consideration of a nonlinear dynamical systems. The High Level CPWL (HL CPWL) formulation used in this paper (Julián *et al.*, 1999) is based on a simplicial partition of the input domain such that the system has a linear affine formulation in each simplex which is continuous on the boundaries.

Under slowly varying assumptions, different linear controller scheduling techniques have been proposed in the literature ((Rugh and Shamma, 2000), (Shamma and Athans, 1991), (Shamma and Athans, 1992), (Galán *et al.*, 2004)).

Within the context of fuzzy logic and neural networks, the controller scheduling idea has received the attention of researchers ((Palm and Stutz, 2003), (Chen and Huang, 2004)) as well.

In the framework of these ideas, in this paper we present a new local linear control strategy of a nonlinear system based on a CPWL NOE model description. In ((Castro *et al.*, 2005*b*), (Castro *et al.*, 2005*a*)) a CPWL Nonlinear Output Error (NOE) model structure and an identification algorithm were presented. This structure is similar to the one proposed by Narendra and Parthasarathy

[1] Partially supported by the ANPCyT
[2] Partially supported by CONICET
[3] Partially supported by CIC

(Narendra and Parthasarathy, 1990) in the context of Neural Networks. The CPWL NOE structure uses HL CPWL functions to develop an identification algorithm that offers a simple mechanism for increasing the model approximation capabilities, retaining the approximation achieved when moving from a coarse grid to a finer one. In this way, it is possible to start the identification with a linear approximation and then increase the model's degrees of freedom progressively in order to reduce the mismatch up to an acceptable value. On the other hand, a reduced model may be evaluated to alleviate overfitting. It is also important to remark that the NOE algorithm assures minimum noise effect in the identified model.

The HL CPWL formulation of the NOE model allows to follow in real time the simplex where the system is actually situated. Each simplex is directly related with the corresponding linear model. Then, under invertibility assumptions, *i.e.* minimum phase assumption of all linear models, a simple controller composed of the local inverse linear model in cascade with a filter to tune the performance, is considered. The minimum phase assumption can be relaxed since special consideration may be applied to guarantee stability of the closed loop system. The particular controller formulation is used only to present a general idea of controller scheduling using CPWL NOE model formulation.

The paper is organized as follows. In Section 2 the identification algorithm (Castro *et al.*, 2005*a*) is reviewed. In Section 3 the proposed control scheme is discussed; in Section 4 we develop an example using the proposed methodology. Finally, in Section 5 we draw some conclusions.

2. IDENTIFICATION METHODOLOGY

Let $(\mathbf{u}, \mathbf{y})$ the input/output available data of length L corresponding to a given Lipschitz continuous, SISO system. If $\tilde{\mathbf{y}}$ is the estimated value corresponding to the input $\mathbf{u}$, and

$$\mathbf{u}^{k,M+1} = [u_k, \, \ldots, u_{k-M}] \quad (1)$$
$$\tilde{\mathbf{y}}^{k-1,N} = [\tilde{y}_{k-1}, \ldots, \tilde{y}_{k-N}], \quad (2)$$

we propose the following black-box identification structure

$$\begin{aligned}\tilde{y}_k &= f_{pwl}\left(u_k, \, \ldots, u_{k-M}, \tilde{y}_{k-1}, \ldots, \tilde{y}_{k-N}\right) \\ &= \mathbf{c}\Lambda\left([\mathbf{u}^{k,M+1}, \tilde{\mathbf{y}}^{k-1,N}]\right). \quad (3)\end{aligned}$$

where the the HL CPWL function $f_{pwl}(\mathbf{x}) = \mathbf{c}\Lambda(\mathbf{x})$ is defined as in (Julián *et al.*, 1999), (Julián, 2000) and the model orders M and N are given. This identification structure, pictured in Fig. 1, can be considered a black box model where the regression vector noted by $\boldsymbol{\varphi}^k$ is taken as $\boldsymbol{\varphi}^k = \left[\mathbf{u}^{k,M+1}, \tilde{\mathbf{y}}^{k-1,N}\right]$ see (Sjöberg *et al.*, 1995), for example). It is worth to mention that a linear model is a particular case of f_{pwl}. Finally, let us note

$$\mathbf{z}^k = \left[\, \mathbf{u}^{k,M+1}, \; \tilde{\mathbf{y}}^{k-1,N} \,\right]. \quad (4)$$

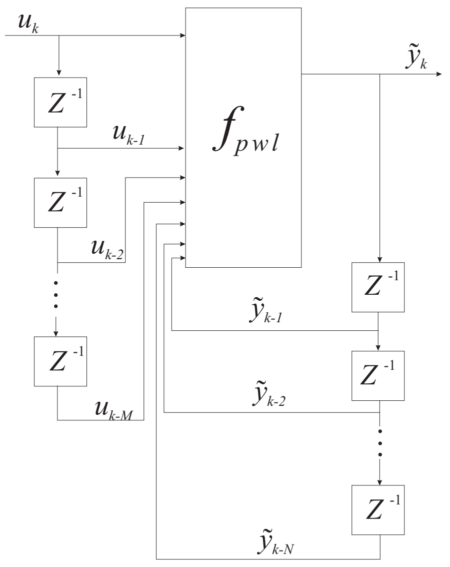

Fig. 1. CPWL NOE model

The domain of the function f_{pwl} is a compact set $\mathbf{S} \subset \mathbb{R}^m, m = M + N + 1$, defined as follows

$$\begin{aligned}\mathbf{S} = \{\mathbf{z} \in \mathbb{R}^m : \underline{z}_i \leq z_i \leq \overline{z}_i; \; \overline{z}_i = \underline{z}_i + \delta.ndiv, \\ i = 1, 2, \ldots, m\}, \quad (5)\end{aligned}$$

being δ the fixed grid size, $\underline{z}_i = \min z_i$ and $\overline{z}_i = \max z_i$ over the entire input/output set with $\mathbf{z}$ defined in (4).

According to ((Julián *et al.*, 1999), (Julián, 1999)), the set defined by (5) is partitioned into polyhedral regions using a simplicial boundary configuration and f_{pwl} is linear on each simplex and continuous on the adjacent boundaries of the simplices.

Taking into account Eq. (5) each dimension is divided into a number of subintervals of equal length δ. Then, when the grid size δ decreases, the number of divisions $ndiv$ on each direction increases. As a consequence, using HL CPWL functions for the nonlinear approximation, $ndiv$ allows to go from a linear model ($ndiv = 1$) with a coarse partition of the domain to a nonlinear one with a finer partition of $\mathbf{S}$. The advantages of using this kind of models is pointed out in (Sjöberg and Ngia, 1998, Ch.1).

According to the proposed methodology we write the identification algorithm using the following notation.

Notation

$ndiv = 2^d, d \geq 0$: number of divisions of the region $\mathbf{S}$. Equal number of divisions in each dimension is assumed.

V^d: the set of vertices of the simplicial partition of the set $\mathbf{S}$ with $ndiv = 2^d$ number of divisions.

Λ^d: The HL CPWL basis defined on $\mathbf{S}$ with vertices belonging to V^d.

$\mathbf{c}^{d,*}$: the row vector of parameters associated with the basis Λ^d. The number of parameters is $(ndiv+1)^{M+1+N}$ ((Julián *et al.*, 1999), (Julián, 1999)).

$(A)_j$: the j-th row of a matrix A.

$Niter \in \mathbb{R}$: maximum number of iterations of the optimization algorithm.

$Maxerror$: maximum allowable approximation error.

$\mathbf{lr}$: learning rate, $lr_i > 0\, \forall i$ ($lr_i = 0.001$).

mom: momentum, $mom > 0$ ($mom = 0.9$).

lr_{inc}: learning rate increment, $lr_{inc} > 1$ (typically, $lr_{inc} = 1.05$).

lr_{dec}: learning rate decrement, $0 < lr_{dec} < 1$ (typically, $lr_{dec} = 0.3$).

η: constant update, $0 \leq \eta \leq 1$.

Identification Algorithm

Step 1. **$d = 0$:** *Linear Approximation.*
Compute the set of parameters $\mathbf{c}^{d,*}$ of the linear model solving the following LS problem

$$\begin{aligned}\mathbf{c}^{d,*} &= \min\left(\|\mathbf{y} - \tilde{\mathbf{y}}\|^2\right) \\ &= \arg\left\{\min_{\mathbf{c}^d}\left\{\frac{1}{2}\sum_{i=1}^{L}\left[y_i - \right.\right.\right. \\ &\qquad \left.\left.\left.\mathbf{c}^d\Lambda^d\left([\mathbf{u}^{i,M+1}, \tilde{\mathbf{y}}^{i-1,N}]\right)\right]^2\right\}\right\}.\end{aligned}$$

Step 2. **$d \leftarrow d + 1$: .** Set $r = 0$. Evaluate the initial condition $\mathbf{c}^{d,*}$ for the new d according to the algorithm given in (Castro *et al.*, 2005*a*). Set

$$\begin{array}{lll} r = 0, & \mathbf{c}^{d,r} = \mathbf{c}^{d,*}, & \eta = 0, \\ \Delta\mathbf{c}^{d,r} = [0, \ldots, 0] & lr_i^0 = 0.001\, \forall i. & \end{array}$$

Step 3. **$r \leftarrow r + 1$:** *Error and gradient evaluation.*

$$E^r = \frac{1}{2}\sum_{i=1}^{L}\left[y_i - \mathbf{c}^{d,r-1}\Lambda^d\left([\mathbf{u}^{i,M+1}, \tilde{\mathbf{y}}^{i-1,N}]\right)\right]^2 \tag{6}$$

$$\begin{aligned}\nabla E_j^r &= \frac{\partial E^r}{\partial c_j^{d,r-1}} \\ &= -\sum_{i=1}^{L}\left[y_i - \mathbf{c}^{d,r-1}\Lambda^d\left([\mathbf{u}^{i,M+1}, \tilde{\mathbf{y}}^{i-1,N}]\right)\right]. \\ &\qquad .\left(\Lambda^d\left([\mathbf{u}^{i,M+1}, \tilde{\mathbf{y}}^{i-1,N}]\right)\right)_j.\end{aligned}$$

Step 4. Parameter update. If $E^r \leq Maxerror$ then STOP; otherwise

$$\Delta c_j^{d,r} = \eta\left(-\nabla E_j^r lr_j^r + \Delta c_j^{d,r-1} mom\right), \tag{7}$$

$$\mathbf{c}^{d,r} = \mathbf{c}^{d,r-1} + \Delta\mathbf{c}^{d,r}, \tag{8}$$

where the constant η is precisely defined in Appendix B and the components of the learning rate vector $\mathbf{lr}^r$ are modified as described below.

$$lr_j^r = \begin{cases} lr_j^{r-1} \times lr_{inc} \text{ if } \operatorname{sign}\left(\nabla E_j^r\right) = \operatorname{sign}\left(\nabla E_j^{r-1}\right) \\ lr_j^{r-1} \times lr_{dec} \text{ if } \operatorname{sign}\left(\nabla E_j^r\right) \neq \operatorname{sign}\left(\nabla E_j^{r-1}\right). \end{cases}$$

If $r < Niter$ go to Step 3.
else
$\mathbf{c}^{d,*} = \mathbf{c}^{d,r}$, go to Step 2.

Remark 1. In order to improve the algorithm performance, any of the well known stop conditions based on the error evolution, may be applied in Step 4.

Remark 2. The described algorithm could be modified to reduce the order of the model. Then, the solution at any level could be found backwards. If only the solution before the last one must be recovered, other possibility would be to retain that vector of parameters and reconstruct the solution using the expression $\tilde{}_k = \mathbf{c}^{d,r}\Lambda^d\left([\mathbf{u}^{k,M+1}, \tilde{\mathbf{y}}^{k-1,N}]\right)$ for all k, $1 \leq k \leq L$.

The advantages and drawbacks of this algorithm have been pointed out in ((Castro *et al.*, 2005*b*), (Castro *et al.*, 2005*a*)).

3. CONTROLLER DESIGN

In order to design the controller, we use the fact that in each sector of the domain, the CPWL NOE structure behaves as a linear model. Then, we will use a linear controller for each one of these sectors.

To design these linear controllers, we adopt the direct synthesis approach (Ogunnaike and Ray, 1994). The controller specification produces a feedback system with a closed-loop pole in a_c without offset, when the set point is changed in the form of steps. If the discrete transfer function of the linear model is called $H(z)$, and $H^{-1}(z)$ is stable and causal, then the controller can be described as

$$K(z) = \frac{a_c}{z-1}\frac{1}{H(z)}. \tag{9}$$

Then, the controller algorithm involves two steps, first it is necessary to extract the linear model valid for the current operating point; then, with this model, a controller is designed.

Let us consider that the process is actually in a given operating point described by the vector $\left[\mathbf{u}^{k,M+1}, \tilde{\mathbf{y}}^{k-1,N}\right]$. Then it is simple to determine the simplex $R^{(i)}$ that contains this point, for example using the PWL Toolbox of MATLAB

(Julián, 2000). Note that if the dimension of the domain is $m = N + M + 1$ each simplex is defined by their $m + 1$ vertices. Then the gradient $J^{(i)}$ in this simplex can be easily evaluated using the values of f_{pwl} at its vertices (see Appendix A). The parameters of the linear model at the region are the entries of this gradient vector. Then it is possible to implement the control algorithm (see Fig. 2) as follows.

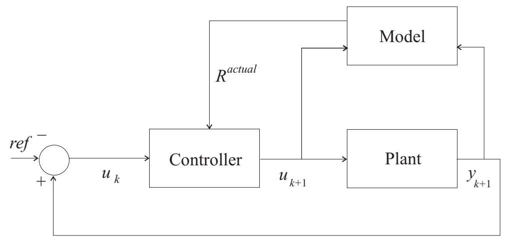

Fig. 2. Control scheme.

Control Model Algorithm

Data. At time k, the past values of the measured and controlled variables to form $\mathbf{x} = \left[\mathbf{u}^{k,M+1}, \tilde{\mathbf{y}}^{k-1,N}\right]$. The desired closed loop time constant (a_c).
Step 1. Determine the sector R^{old} in which is operating the process, the linear model corresponding to this sector (using the algorithm of Appendix B) and the linear controller K^{actual} of Eq. (9).
Step 2. Determine the actual sector R^{actual}.
Step 3. If R^{actual} is not equal to R^{old}, compute the actual linear model, the new controller K^{actual} and make $R^{old} = R^{actual}$.
Step 4. Apply the corresponding manipulated variable u_k to the process, and measure y_k.
Step 5. Compute the new manipulated variable u_{k+1} (computed using K^{actual} and y_k). Make $k \leftarrow k + 1$.
Step 6. Update $\mathbf{x} = \left[\mathbf{u}^{k-M}, \mathbf{y}^{k-N}\right]$ using u_{k+1} and y_k and return to Step 2.

Remark 3. It is obvious that the performance of this control algorithm depends on the quality of the model. In this case, we consider that each simplex in the partition of the domain have enough data to allow a good quality model. In this way, it is possible to improve the robustness of the control algorithm by relaxing the control specification *i.e.* the time constant when the number of data in a given region is small.

4. EXAMPLE

In this example we model the neutralization reaction between a strong acid and a strong base in the presence of a buffer agent as described by (Galán, 2001) (for a complete description of this process and a first principles models see (Biagiola *et al.*, 2004)).

The goal is to control the output pH, by manipulating the alkaline solution flow rate q_B. The operating point for the neutralization is $q_B = 0.5$ and $pH = 7.7182$.

In order to identify the system, q_B is excited by a random signal with uniform distribution between 0 and 1, the limits of the physical variable. Time simulation is performed for a 250 seconds for a sample time of 0.25 sec. The regression vector of the CPWL NOE model is taken as $\varphi_k = [u_k, \tilde{y}_{k-1}, \tilde{y}_{k-2}]$ and the number of divisions for each variable are two, four and eight, giving a total number of parameters equal to 27, 125 and 729, respectively. The parameters of the algorithm for adjusting the vector of parameters are taken as $\mathbf{lr}^0 = 0.0002$, $mom = 0.9$, $inc = 1.05$ and $dec = 0.3$.

In Fig. 3 the error is displayed as a function of the iterations; the number of divisions of the simplices increases every 1000 iterations.

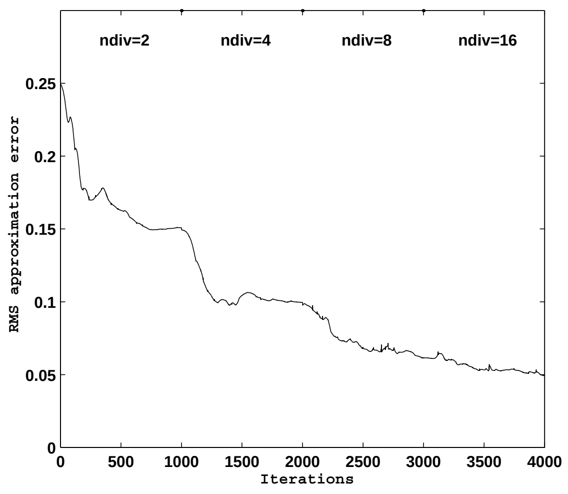

Fig. 3. RMS error for pH neutralization.

The controller is used to follow a set point change. The controller parameter is set at $a_c = 0.5$. The simulation for this control is shown in Fig. 4. In this figure, the system response for the controller with $ndiv = 2, 4$ and 8, $ndiv = 4$ are shown. Again, from these plots it is clear that the controller performance improves when the number of divisions increase.

5. CONCLUSIONS

In this paper, a NOE identification algorithm based on HL CPWL functions approximation method is reviewed, and an algorithm to control this model structure is presented. The identification methodology allows to approximate a NOE

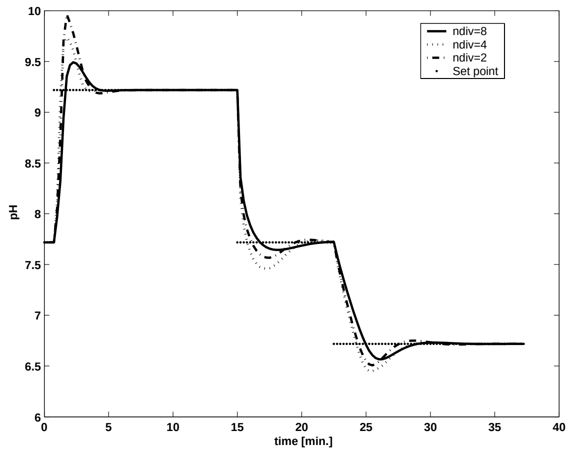

Fig. 4. Simulation for pH neutralization control.

model from a linear one, and the control scheme uses the linear information in each region of operation to design a simple controller. The main feature of this process is that it enables to go from a linear model to a nonlinear one straightforwardly. Finally, the potentials of our approach have been illustrated with two examples.

REFERENCES

Biagiola, S.I., O.E. Agamennoni and J.L. Figueroa (2004). H_∞ control of a Wiener type system. *International Journal of Control* **77**(6), 572–583.

Castro, L.R., J.L. Figueroa and O.E. Agamennoni (2005*a*). BIBO stability for NOE model structure using HL CPWL functions. In: *Proc. of the 24th IASTED Int. Conf., MIC 2005.* Univ. California, Berkeley. pp. 91–96.

Castro, L.R., J.L. Figueroa and O.E. Agamennoni (2005*b*). An NIIR structure using HL CPWL functions. *Latin Amer. Appl. Res.* **35**, 161–166.

Chen, J. and T-C. Huang (2004). Applying neural networks to on-line updated PID controllers for nonlinear process control. *Joural of Process Control* **14**, 211–230.

Galán, O. (2001). Robust multi-linear model*-based control for nonlinear plants. PhD thesis. The University of Sydney. Sydney, Australia.

Galán, O., J.A. Romagnoli and A. Palazoglu (2004). Real-time implementation of multi-linear model-based control strategies: an application to a bench-scale pH neutralization reactor. *Journal of Process Control* **14**, 571–579.

Julián, P. (2000). A toolbox for the piecewise linear approximation of multidimensional functions. http://www.pedrojulian.com.

Julián, P., A. Desages and O. Agamennoni (1999). High level canonical piecewise linear representation using a simplicial partition. *IEEE Trans. on Circ. and Syst.* **44**, 463–480.

Julián, Pedro M. (1999). A High Level Canonical Piecewise Linear Representation: Theory and Applications. PhD thesis. Universidad Nacional del Sur, Bahía Blanca, Argentina. UMI Dissertation Services, Michigan, USA.

Narendra, K. S. and K. Parthasarathy (1990). Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks* **1**, 4–27.

Ogunnaike, B.A. and W.R. Ray (1994). *Process Dynamics, Modeling and Control.* Oxford University.

Palm, R. and C. Stutz (2003). Open loop dynamic trajectory generator for a fuzzy gain scheduler. *Engineering Applications of Artificial Intelligence* **16**, 213–225.

Rugh, W. and J. Shamma (2000). Research on gain scheduling. *Automatica* **36**, 1401–1425.

Shamma, J. and M. Athans (1991). Guaranteed properties of gain scheduled control for linear parameter-varying plants. *Automatica* **27**, 559–564.

Shamma, J. and M. Athans (1992). Gain scheduling: potential hazards and possible remedies. *IEEE Control Systems Magazine* **12**, 101–107.

Sjöberg, J. and L. S. H. Ngia (1998). *Neural Nets and Related Model Structures for Nonlinear System Identification.* pp. 1–28. Nonlinear Modeling: Advanced Black-Box Techniques. J. A. K. Suykens and J. Vandevalle, Eds.. Kluwer Academic Publishers.

Sjöberg, J., Q. Zhang, L. Ljung, A. Beneviste, B. Delyon, P. Glorennec, H. Hjalmarsson and A. Juditsky (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica* **35**(12), 1691–1724.

Appendix A. LINEAR MODEL FROM CPWL FUNCTIONS

Let us consider a HL CPWL function $f_{pwl} : \mathbf{S} \subset \mathbb{R}^m \rightarrow \mathbb{R}^1$ and a vector $\mathbf{x}$ at a given region $R^{(k)}$, *i.e.* $\mathbf{x} \in R^{(k)} \subset \mathbf{S}$. This region $R^{(k)} \in \mathbb{R}^m$ is uniquely defined by its $m+1$ vertices $V^{(j)}, j = 1, \ldots, m+1$. Once that these vertices are computed [4], it is possible to obtain the linear model $J^{(j)} \in \mathbb{R}^m$ that represents the process at this region. An algorithm to perform the linear model can be written as follows.

Linear Model Algorithm

Step 1. Compute the value of f_{pwl} at vertices,

[4] This is trivial using the function *vertices.m* from (Julián, 2000)

```
        for $i = 1$ to $m + 1$
            $(z)_i = \mathbf{c}\Lambda\left(V^{(i)}\right)$,
        end
Step 2. From the vector and matrix,
        for $i = 1$ to $m$
            $(\Gamma)_i = (z)_i - (z)_{m+1}$,
            $\Theta(:, i) = V^{(i)} - V^{(m+1)}$,
        end
    where $\Theta(:, i)$ represents the $i^{th}$ column of the
    $\Theta$ matrix.
Step 3. Compute the gain vector $J$ ,
        $J = \Theta^{-1}\Gamma$
```

Appendix B. BIBO STABILITY

Let us suppose that $\mathbf{u} \in U \subset \mathbb{R}^{M+1}$, $\mathbf{y} \in O \subset \mathbb{R}^N$, U and O given compact sets, $Q \subset U \times O$, Q compact and $I = \left[\underline{y}, \overline{y}\right] \subset \mathbb{R}$, with $\underline{y} = \min \mathbf{y}$, $\overline{y} = \max \mathbf{y}$.

Definition 4. We say that the model defined by (3) is BIBO stable if $f_{pwl}(Q) \subset I$.

This definition means that the model output remains within the output values when the input is any signal $\mathbf{u} \in U$.

The expression (3) defines a mapping $f_{pwl} : Q \to I$. As Q is a compact set and f_{pwl} is continuous on Q, then it attains its maximum and minimum values on Q. Moreover, since f_{pwl} is linear on each simplex, the extreme values are attained on V_Q, the set of vertices of Q. Then the NOE identification structure given by (3) will be BIBO stable if the minimization problem

$$\min_{\mathbf{c}^d} E^r \text{ s.t. } \max_{\substack{\mathbf{z}^k \in Q \\ 1 \le k \le L}} f_{pwl}\left(\mathbf{z}^k\right) = \max_{\mathbf{v} \in V^d} \left|\mathbf{c}^d \Lambda(\mathbf{v})\right| \subset I,$$

where E^r is given by equation Eq. (6) and V^d is defined in Section 2, is solvable. This minimization problem is equivalent to

$$\min_{\mathbf{c}^d} E^r \text{ s.t.} \begin{cases} \max\limits_{\mathbf{v} \in V^d} \left(\left|\mathbf{c}^d \Lambda(\mathbf{v})\right|\right) \le \overline{y} \\ \min\limits_{\mathbf{v} \in V^d} \left(\left|\mathbf{c}^d \Lambda(\mathbf{v})\right|\right) \ge \underline{y}. \end{cases} \quad \text{(B.1)}$$

Remark 5. Considering a 10.000-length input vector, it takes around $16\mu s$ of CPU to solve the minimization problem with restrictions given by (B.1) in the MATLAB environment, with a Pentium IV, 512Mb RAM computer.

Once we have found $\mathbf{c}^d$ that satisfies (B.1), we must guarantee that, for any r with d fixed, is $\tilde{\mathbf{y}}_k = \mathbf{c}^{d,r}\Lambda\left(\mathbf{z}^k\right) \in I, 1 \le k \le L$, where $\mathbf{c}^{d,r}$ is obtained from Step 4 of the identification algorithm described in Section 2.

We can state the following sufficient condition.

Proposition 1. Let us suppose that, for d and $r-1$ fixed, the model is BIBO stable. Then the model will be BIBO stable for d and r if the following condition is satisfied

$$\begin{aligned} \underline{y} - \min_{\mathbf{v} \in V_d} \left(\mathbf{c}^{d,r-1}\Lambda(\mathbf{v})\right) &\le \Delta\mathbf{c}^{d,r}\Lambda(\mathbf{v}) \\ &\le \overline{y} - \max_{\mathbf{v} \in V_d} \left(\mathbf{c}^{d,r-1}\Lambda(\mathbf{v})\right), \end{aligned} \quad \text{(B.2)}$$

where $\Delta\mathbf{c}^{d,r}$ is given by equation Eq. (8) and $\mathbf{v} \in V^d$.

Proof. See (Castro *et al.*, 2005*a*).

Corollary 2. With the hypothesis of Proposition 1, let us note $a = \underline{y} - \min_{\mathbf{v} \in V^d}\left(\mathbf{c}^{d,r-1}\Lambda(\mathbf{v})\right)$, $b = \overline{y} - \max_{\mathbf{v} \in V^d}\left(\mathbf{c}^{d,r-1}\Lambda(\mathbf{v})\right)$. Then

$$\eta \ge \frac{a}{\min_{\mathbf{v} \in V^d}\left[\left(-\nabla E_j^r lr_j^r + \Delta c_j^{d,r-1} mom\right)\Lambda(\mathbf{v})\right]}, \quad \text{(B.3)}$$

$$\eta \le \frac{b}{\max_{\mathbf{v} \in V^d}\left[\left(-\nabla E_j^r lr_j^r + \Delta c_j^{d,r-1} mom\right)\Lambda(\mathbf{v})\right]} \quad \text{(B.4)}$$

simultaneously.

Proof. See (Castro *et al.*, 2005*a*).

Remark 6. From both bounds for η given by Eq. B.3 and B.4, the only one with practical interest is the least positive one and is the bound used in Step 4 of the identification algorithm.