2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)
July 6-11, 2014, Beijing, China

Brain Style Control scheme: Simultaneous Forward and Inverse Model identification and Controller design

Luka Eciolaza*, Tanadari Taniguchi†, Michio Sugeno*
*European Centre for Soft Computing, Mieres, Asturias, Spain
Email: luka.eciolaza@softcomputing.es, michio.sugeno@gmail.com
†Tokai University, Hiratsuka, Kanagawa, 2591292 Japan
Email: taniguchi@tokai-u.jp

***Abstract*—Brain Style control scheme is presented inspired by the brains hability to implement the most efficient and robust control systems available to date. It represents the first attempt to perform simultaneous and on-line (a) forward model identification, (b) controller design and (c) inverse model learning. For that we have simultaneously implemented the concepts of *Vertex Placement Principle* and *Feedback Error Learning*. The paper contributes in our attempt to demonstrate the potential of Piecewise Bilinear models for nonlinear control systems.**

***Keywords*—Brain style control, Piecewise bilinear models, Vertex placement principle, Feedback error learning, Nonlinear control systems.**

I. Introduction

The brain implements the most efficient and robust control system available to date. Works in adaptive control and artificial neural networks were essential in understanding the function of the brain [1], which provided valuable insights for the design of efficient learning and control systems. Thus, the concept of an internal model has emerged as an important theoretical concept in motor control [2]. There are two types of internal model, forward and inverse models.

Forward models are important in sensory perception of the brain. They capture the forward or causal relationship between inputs to the system, e.g. the arm, and the outputs. A forward dynamic model of the arm, for example predicts the next state (e.g. position and velocity) given the current state and motor command. When sensing the state of the system cannot be done directly, or has a big delay, it is useful to predict the current state using the internal model of the system dynamics. Such models have been proposed to be used in motor learning, state estimation and motor control. Even for a simple arm-reaching movement, a control strategy which takes into account the dynamic property of the arm should be used.

On the other hand, inverse models invert the system by providing the motor command which will cause a desired change in state. Therefore, inverse models act efficiently as controllers as they can provide the motor command necessary to achieve some desired state transition. According to [3], the brain uses inverse dynamical models in order to calculate necessary commands from desired trajectory information. Fast and coordinated limb movements cannot be executed solely under feedback control, since biological feedback loops are slow and have small gains. The inverse model as a feedforward action constitutes an important role for the quick and smooth motions of the limbs in human motor control.

Both forward and inverse models depend on the dynamics of the motor system, which change throughout life and under different contextual conditions, therefore, these models must be adaptable.

In this paper, we implement a *Brain Style* control scheme, where simultaneous (a) plant identification, (b) controller design and (c) inverse model learning, are performed. It is the first time the three tasks are tackled simultaneously and it represents an adaptive control system. If plant and controller are initially unknown it performs an automatic controller design scheme based on a desired plant performance.

Adaptive learning and control systems based on brain functional analysis [1][2][4] only identify forward plant model and its inverse by they do not design a controller. In this paper, we also want to target the control of nonlinear systems and so the task of controller design is considered.

The conventional approach for the analysis of nonlinear control systems [5] [6] are based on local/global linearization methods which transform the analyzed systems into linear models that can then be easily analyzed by linear control theory. Recently, piecewise systems have been studied related to the analysis of nonlinear control systems [7] [8] [9] [10]. The TS systems [11] [12] in fuzzy control, also represent a piecewise method and they aim to approximate general nonlinear systems.

However, our work is based on the uso of piecewise bilinear (PB) approximations proposed by one of the authors in [13]. A bilinear function is a nonlinear bi-affine function: $y = a + bx_1 + cx_2 + dx_1x_2$, the second simplest one after a linear function. This function can represents any four points in the three dimensional space. A PB model is built on rectangular regions with four vertices partitioning the state space. The approximated system is easily applicable for control purpose.

The PB model is a fully parametric model to represent Linear/Nonlinear systems. It is designed to be easily applicable for control purpose and it is a good general approximation of nonlinear systems. The PB models are very convenient for control purpose as they are interpretable (for instance as look up tables), easy to compute and simple to handle. This paper and other recent work we have done [14] [15] [16] [17] [18] [19] are devoted to explore the full potential of PB models for the design of control systems. Thus, all three models used in this paper, namely, forward, inverse and controller, are based on PB models.

The *forward model identification* and *controller design*

978-1-4799-2072-3/14/$31.00 ©2014 IEEE

tasks of the paper are implemented through *Vertex Placement Principle* (VPP) introduced in [16]. VPP represents a systematic approach for simultaneous and on-line global identification and controller design. Even if a plant is unknown initially, we can design a PB controller based on desired plant based on the realized identification. VPP performs an on-line design of a PB-controller C for an unknown plant P based on a desired plant DP so that the closed-loop system CP of the unknown plant and its PB-controller behaves like the desired plant. VPP assumes (a) a PB model of a desired plant is given, and (b) both the states of the desired plant and an unknown plant can be observed and their derivatives can be numerically calculated. It represents a direct approach to design the PB-controller as the design is based directly on the estimation of P. PB model has a big number of parameters to tune, the VPP [16] showed the global tuning of PB models can be made in a very simple and efficient way.

The *inverse model learning* is implemented following a PB model based *Feedback Error Learning* (FEL) presented in [17]. FEL scheme proposed by Kawato [4], represents an inverse model identification scheme. The key idea of FEL is to use the output of the feedback controller as the error signal for the adaptive feedforward controller. FEL was originally inspired on biological control systems which usually are systems with time delays and strong nonlinearities and it has the feature that the learning and control can be performed simultaneously. FEL architecture consists of the combination of a feedback controller that ensures the stability of the system and an adaptive feedforward controller that improves control performance. FEL is able to acquire an accurate inverse model even for redundant systems. In [17] we concluded that PB model is the most convenient model to implement the FEL with regard to nonlinear modeling, control objective and on-line learning capability.

Section II and III introduce the canonical form of PB model, VPP for on-line plant identification and controller design respectively. Section IV presents the PB model based FEL algorithm. Section V presents the implementation issues of the Brain Style control scheme, where a simultaneous VPP and FEL is performed. Section VI illustrates various simulation results, and Section VII concludes the paper with some discussion and future work plans.

II. PB MODELS

The PB model is very convenient for control purpose because is interpretable, easy to compute and simple to handle. In the model, bilinear functions are used to regionally approximate any given function. The obtained model is built on piecewise rectangular regions, and each region is defined by four vertices partitioning the state space. A bilinear function is a nonlinear function of the form $y = a+bx_1+cx_2+dx_1x_2$, where any four points in the three dimensional space are spanned with a bi-affine plane. PB system has a continuous crossing over the piecewise regions and it is a very good general approximator for nonlinear functions. A local error does not trigger a global error and its interpolation nature generates robust outputs. A PB model can be expressed as a look-up table (LUT) which is widely used to realize industrial controllers. Generalization of PB models to a multidimensional case was studied in [13], but for the sake of simplicity here we shall explain a two-dimensional case.

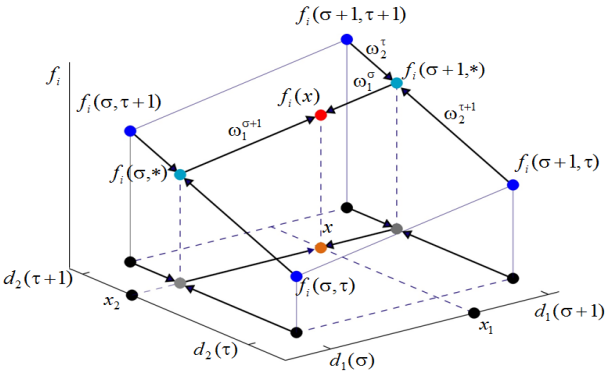

Fig. 1. Piecewise region $R_{\sigma\tau}$ and interpolation of $f_i(x)$.

If a general case of an affine two-dimensional nonlinear control system is considered,

$$\begin{cases} \dot{x}_1 &= f_1(x_1,x_2) \\ \dot{x}_2 &= f_2(x_1,x_2)+g(x_1,x_2)\cdot r \\ y &= h(x_1,x_2) \end{cases} \tag{1}$$

where r is the input (control, reference or both). For the PB representation of a state-space equation, a coordinate vector $d(\sigma,\tau)$ of the state space and a rectangle $\Re_{ij}$ must be defined as,

$$d(i,j) \equiv (d_1(i), d_2(j))^T \tag{2}$$

$$R_{ij} \equiv [d_1(i), d_1(i+1)] \times [d_2(j), d_2(j+1)] \tag{3}$$

where $i \in (1,\ldots,n_1)$ and $j \in (1,\ldots,n_2)$ are integers, and $d_1(i) < d_1(i+1)$, $d_2(j) < d_2(j+1)$. The PB models are formed by matrices of size $(n_1 \times n_2)$, where n_1 and n_2 represent the number of partitions of dimension x_1 and x_2 respectively. Each value in the matrix is referred to as a vertex in the PB model. The operational region of the system is divided into $(n_1-1 \times n_2-1)$ piecewise regions that are analyzed independently.

The PB model was originally derived from a set of fuzzy if-then rules with singleton consequents [13] such that

$$\text{if } x \text{ is } W^{\sigma\tau}, \text{ then } \dot{x} \text{ is } f(\sigma,\tau) \tag{4}$$

which in a two-dimensional case, $x \in \Re^2$ is a state vector, $W^{\sigma\tau} = (w_1^{\sigma}(x_1), w_2^{\tau}(x_2))^T$ is a membership function vector, $f(\sigma,\tau) = (f_1(\sigma,\tau), f_2(\sigma,\tau))^T \in \Re$ is a singleton consequent vector, and $\sigma,\tau \in Z$ are integers ($1 \le \sigma \le n_1, 1 \le \tau \le n_2$) defined by,

$$\sigma(x_1) = d_1(\max(i)) \text{ where } d_1(i) \le x_1, \tag{5}$$

$$\tau(x_2) = d_2(\max(j)) \text{ where } d_2(j) \le x_2. \tag{6}$$

The superscript T denotes transpose operation.

For $x \in \Re_{\sigma\tau}$, the PB models that approximates (1) is expressed as,

$$\begin{cases} f_1(x_1,x_2) = \sum_{i=\sigma}^{\sigma+1}\sum_{j=\tau}^{\tau+1} w_1^i(x_1)w_2^j(x_2)f_1(i,j), \\ f_2(x_1,x_2) = \sum_{i=\sigma}^{\sigma+1}\sum_{j=\tau}^{\tau+1} w_1^i(x_1)w_2^j(x_2)f_2(i,j), \\ g(x_1,x_2) = \sum_{i=\sigma}^{\sigma+1}\sum_{j=\tau}^{\tau+1} w_1^i(x_1)w_2^j(x_2)g(i,j), \\ h(x_1,x_2) = \sum_{i=\sigma}^{\sigma+1}\sum_{j=\tau}^{\tau+1} w_1^i(x_1)w_2^j(x_2)h(i,j), \end{cases} \tag{7}$$

where

$$\begin{cases} w_1^{\sigma}(x_1) &= 1-\alpha, \\ w_1^{\sigma+1}(x_1) &= \alpha, \\ w_2^{\tau}(x_2) &= 1-\beta, \\ w_2^{\tau+1}(x_2) &= \beta, \end{cases} \tag{8}$$

and

$$\alpha = \frac{x_1 - d_1(\sigma)}{d_1(\sigma+1) - d_1(\sigma)} \tag{9}$$

$$\beta = \frac{x_2 - d_2(\tau)}{d_2(\tau+1) - d_2(\tau)} \tag{10}$$

in which case $w_1^i, w_2^j \in [0,1]$.

In every region of the PB models, i.e.: $f_1(x_1, x_2)$, the values are computed through bilinear interpolation of the corresponding four vertexes as shown in Fig. 1. Note that the approximation can be made by only using the values of a nonlinear function at the vertexes of R_{ij}'s in (3). A better approximation of original model can be obtained using *optimal PB modeling* method presented in [15], where a $f(x_1, x_2)$ is generalized as a function of the vertexes included in all local PB models.

III. VPP: On-line Design of LUT Controllers

Given a partition of the sate space of the objective plant P into sub-regions, the PB model is completely described by the values of P at the vertexes of regions. Therefore, to design a LUT-controller means to assign the values of the vertexes. *Vertex Placement Principle* assigns the values of a LUT-controller C at the vertexes, based on a desired plant PB model and an objective plant PB model to be identified. If as in [20], a LUT-controller was assigned so that the values of the closed-loop system at the vertexes satisfy stability conditions, in this sense, VPP is similar to the idea of *Pole Placement* in linear systems.

The algorithm to implement the on-line design of LUT proposed in [16] has three clear steps as shown in Fig. 2. Once the controller C is initialized, either as a table with all zero elements, or some other value if prior knowledge of the plant is available. (a) Then with a reference input r and the observation of current states of an objective plant (P) the estimation of the closed-loop plant (CP) is done. (b) Using a currently estimated CP model and the controller C, P is estimated. (c) Based on currently estimated plant P and the desired plant we update a LUT-controller as $C = DP - P$. Steps (a), (b) and (c) are repeated taking account of control input u and the updated values of controller C.

As we have mentioned above, in this work we want to design a LUT controller to make an initially unknown nonlinear plant (P) type

$$\begin{cases} \dot{x}_{1p} = x_{2p} \\ \dot{x}_{2p} = f_{2p}(x_{1p}, x_{2p}) + u + r \end{cases} \tag{11}$$

behave like a desired plant (DP) given by,

$$\begin{cases} \dot{x}_{1d} = x_{2d} \\ \dot{x}_{2d} = f_{2d}(x_{1d}, x_{2d}) + r \end{cases} \tag{12}$$

The performance of the plant P with the controller C in closed loop CP will be approximated to the performance of

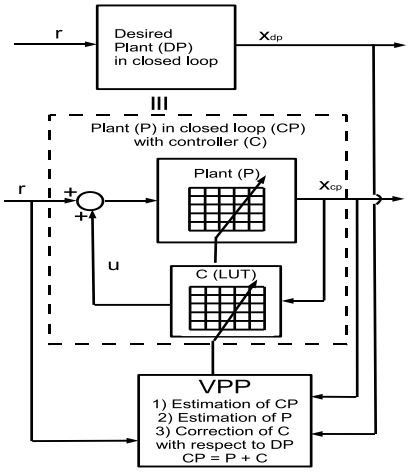

Fig. 2. VPP scheme for on-line plant identification and controller design.

DP, where CP is described by

$$\begin{cases} \dot{x}_{1cp} = x_{2cp} \\ \dot{x}_{2cp} = f_{2cp}(x_{1cp}, x_{2cp}) + r \end{cases} \tag{13}$$

Should the plant to be controlled be known, the closed loop behavior of the plant with a feedback control u would be given by,

$$f_{2cp}(x_{1cp}, x_{2cp}) = f_{2p}(x_{1cp}, x_{2cp}) + u(x_{1cp}, x_{2cp}) \tag{14}$$

and the controller computed directly as,

$$u(x_{1cp}, x_{2cp}) = f_{2cp}(x_{1cp}, x_{2cp}) - f_{2p}(x_{1cp}, x_{2cp}). \tag{15}$$

Even in case that the plant is unknown initially, if we are able to approximate a PB model through an off-line analysis, the computation of $u(x_{1cp}, x_{2cp})$ is straight forward.

However, VPP deals with the on-line design of a LUT controller for an unknown plat. In the relation (14) containing three variables, we can set $f_{2cp} = f_{2d}$ where f_{2d} is given. We still have two unknown variables f_{2p} and u for one relation. In order to solve this problem, we estimate f_{2p}: a PB model corresponding to the unknown plant P, and so, we have a single variable u to solve.

After the initialization of u either as a table with all zero elements, or some other value if prior knowledge of the plant is available. Then an estimation of the PB model that approximates the plant will be performed. At every instant, the plant vertexes corresponding to the region in which the plant is operating will be estimated. The values of control u at the vertexes of the same operational region will be updated according to the plant estimation and based on the vertex placement principle.

From the plant in closed loop CP, at every instant k we can measure its state variables ($x_{1cp}(k)$, $x_{2cp}(k)$), their derivatives ($\dot{x}_{1cp}(k)$, $\dot{x}_{2cp}(k)$), and the reference input $r(k)$. From here, we can also compute ($f_{2cp}(k)$) as,

$$f_{2cp}(k) = \dot{x}_{2cp}(k) - r(k). \tag{16}$$

The estimation of the vertexes, corresponding to the region in which the plant is operating, is performed only using the time instants when the plant was

inside that given region. The analyzed region of the closed-loop system is delimited by the vector $V_{CP_2} = (V_{CP(\sigma,\tau)}, V_{CP(\sigma,\tau+1)}, V_{CP(\sigma+1,\tau)}, V_{CP(\sigma+1,\tau+1)})^T$ that contains the four vertexes. At every instant, f_{2cp} is expressed by a piecewise bilinear interpolation as shown at Fig. 1, where k implies a step in updating in our algorithm.

$$f_{2cp}(k) = (w_1^\sigma w_2^\tau, w_1^\sigma w_2^{\tau+1}, w_1^{\sigma+1} w_2^\tau, w_1^{\sigma+1} w_2^{\tau+1}) \\ (V_{CP(\sigma,\tau)}, V_{CP(\sigma,\tau+1)}, V_{CP(\sigma+1,\tau)}, V_{CP(\sigma+1,\tau+1)})^T, \tag{17}$$

where w_1^i and w_2^j are defined in (8).

To be precise, (17) is expressed as

$$f_{2cp}(k) = a(x_{cp}(k)) V_{CP_2}(k), \tag{18}$$

Taking an account of updating values of the LUT controller C, the vertexes of the closed loop plant f_{2cp} can be written as,

$$V_{CP_2}(k) = V_{P_2} + V_C(k). \tag{19}$$

which is derived from the relation (14).

Similarly to V_{CP_2}, the vectors V_{DP_2}, V_{P_2} and V_C represent the vertexes defining a region of the desired plant, objective plant and controller respectively. Through VPP, V_C is obtained such that $V_{DP_2} = V_{CP_2} = V_{P_2} + V_C$, where V_{P_2} is initially unknown and need to be estimated, and V_C will be initialized with a known valued, usually an all zero matrix. Thus, from (18) we derive,

$$f_{2cp}(k) - a(x_{cp}(k)) V_C(k) = a(x_{cp}(k)) V_{P_2}. \tag{20}$$

If there are m samples of the system within a given region, we can set the following equation:

$$F_2(\dot{x}_{2cp}, r) - A(x_{cp}) \odot V_C = A(x_{cp}) V_{P_2}, \tag{21}$$

where F_2 is $(m \times 1)$ vector $F_2 = [f_{2cp}(1) \dots f_{2cp}(m)]^T$, A is a $(m \times 4)$ matrix $A(x_{cp}) = [a(x_{cp}(1)), a(x_{cp}(2)), \dots, a(x_{cp}(m))]^T$.

$A(x_{cp}) \odot V_C$ corresponds to the row-sums of their Hadamard product $A(x_{cp}) \odot V_C = \sum (A(x_{cp}) o V_C)$.

From (21) we can obtain the values of V_{P_2} as,

$$(A^T A)^{-1} A^T (F_2 - A \odot V_C) = V_{P_2}, \tag{22}$$

where $A^T A$ is assumed to be non-singular.

Once the V_{P_2} values have been estimated, we can compute the error in each vertex of the region with respect to the desired plant vertexes V_{DP_2},

$$e_{(i,j)}(k) = V_{DP_2(i,j)} - (V_{P_2(i,j)} + V_{C(i,j)}(k-1)). \tag{23}$$

Note VPP computes the error directly for each of the vertexes within the current operating region. Tracking errors $e = x_d - x_p$ or $\dot{e} = \dot{x}_d - \dot{x}_p$ are not used as in model following or adaptive control.

Finally in each instant, the vertexes of the controller C (V_C) will be updated as,

$$V_{C(i,j)}(k) = V_{C(i,j)}(k-1) + w_1^i w_2^j e_{(i,j)}(k). \tag{24}$$

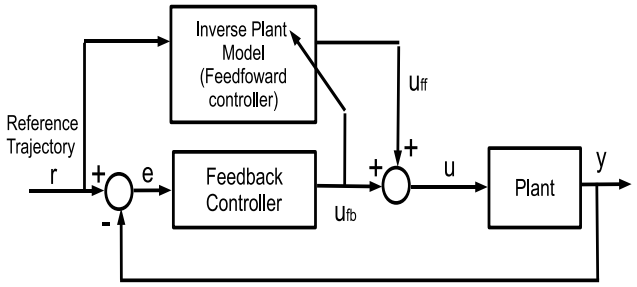

Fig. 3. Feedback error learning architecture.

IV. FEL: Sequential learning

Fig. 3 illustrates the feedback error learning architecture. This architecture consists of an objective plant to be controlled with a feedback controller and, in addition, a feedforward controller to be learned by FEL. In [17] PB model showed to be a very convenient model to implement the control scheme.

If a plant model is expressed as $y = p(u)$ where y is output and u control input. We also denote its inverse in a similar manner as $u = p^{-1}(y)$, assuming that a plant is invertible. We note that an inverse model used in FEL is usually written as $u_{ff} = p^{-1}(r)$. For the PB based FEL, feedforward controller is expressed as $u_{ff} = p^{-1}(r, \dot{r})$, with two inputs r, $\dot{r}$ as desired output of the objective plant and single output u_{ff}.

Let u_0 be an ideal feedforward control based on a inverse model. We design a feedforward controller so that $u_{ff} = u_0$. That is, we learn u_{ff}, i.e., identify the feedforward controller parameters, to minimize the performance index

$$I = \frac{(u_{ff} - u_0)^2}{2} \tag{25}$$

where the PB representation of the feedforward controller is,

$$u_{ff} = p^{-1}(r, \dot{r}) = \sum_{i=\sigma}^{\sigma+1} \sum_{j=\tau}^{\tau+1} w_1^i(r) w_2^j(\dot{r}) V_{(i,j)} \tag{26}$$

I can be sequentially minimized using the derivative of (25)

$$\frac{\partial I}{\partial V} = \frac{\partial u_{ff}}{\partial V}(u_{ff} - u_0). \tag{27}$$

However, the error $(u_{ff} - u_0)$ is not available since u_0 is unknown. Therefore Kawato [4] suggested to use u_{fb} for $(u_{ff} - u_0)$ since $u = u_{fb} + u_{ff}$. This is why u_{fb} is called a feedback error playing a role as the error $(u_{ff} - u_0)$. FEL is a learning scheme based on a signal u_{fb}, a feedback error signal.

Then we have,

$$\frac{\partial I}{\partial V} = \frac{\partial u_{ff}}{\partial V} u_{fb} = \frac{\partial p^{-1}(r, \dot{r})}{\partial V} u_{fb}. \tag{28}$$

The sequential learning of each vertex of a region is made using the following algorithm:

$$V_{new(i,j)} = V_{old(i,j)} - \delta \frac{\partial u_{ff}}{\partial V(i,j)} u_{fb} \tag{29}$$

where δ is an adjustable parameter as a learning rate. This is the conventional steepest descent algorithm to minimize

a performance index. If learning is successfully completed, i.e., $V_{new} = V_{old}$, then u_{fb} must become zero, and only u_{ff} works.

In the case of a two dimensional PB model, if we develop (26), with (8) (9) and (10), we have:

$$\begin{aligned} u_{ff} = (1-\alpha)(1-\beta)V_{(\sigma,\tau)} + (1-\alpha)\beta V_{(\sigma,\tau+1)} \\ +\alpha(1-\beta)V_{(\sigma+1,\tau)} + \alpha\beta V_{(\sigma+1,\tau+1)} \end{aligned} \quad (30)$$

(V_{new}, V_{old}) refer to the values of 4 vertexes of a region and as the function is linear, the calculation of partial derivatives (∇p^{-1}) is straightforward. The equation of change for each vertex of a region is,

$$V_{new(i,j)} = V_{old(i,j)} - \delta w_1^i w_2^j u_{fb}, \quad (31)$$

where w_1^i and w_2^j are defined in (8).

Note that in [17] an off-line identification of pseudo-inverse model is also proposed. This method uses I/O data to make an approximation of the inverse model and it is mainly used for initialization of the feedforward model within the FEL scheme.

V. Brain Style Control scheme based on PB models: Simultaneous VPP and FEL

All the models used in this paper are based on piecewise bilinear (PB) models. The three different but simultaneous tasks performed in the *Brain Style* control scheme, (a) plant identification, (b) controller design and (c) inverse model learning, are represented by PB models.

As we have seen in previous sections, the PB model is a fully parametric model to represent Linear/Nonlinear systems. It is designed to be easily applicable for control purpose and it is a good general approximation of nonlinear systems. This papers extends on our work to explore the potential of PB models.

The *plant identification* and *controller design* is implemented through VPP introduced in [16]. VPP represents a systematic approach for simultaneous and on-line global identification and controller design. Even if a plant is unknown initially, we can design a PB controller based on desired plant based on the realized identification.

The *inverse model learning* is implemented following a PB model based FEL presented in [17]. A method to make an on-line sequential learning of a feedforward controller as a PB model was presented. Moreover, we think PB model is the most convenient model to implement FEL with regard to nonlinear modeling, control objective and on-line learning capability.

Fig. 4 shows the schematic diagram of the *Brain Style Control* scheme proposed in this paper. It is a combination of both VPP and FEL schemes presented before and it also includes a feedback controller block that tries to correct the tracking error $e = r - y$.

Apart from the PB model parameters to be adjusted for which VPP and FEL algorithms have been presented. This control scheme needs to tune various parameters, such as, the learning rate δ of the FEL.

On the other hand, the selection of the feedback controller is also important. From experimentation we have observed

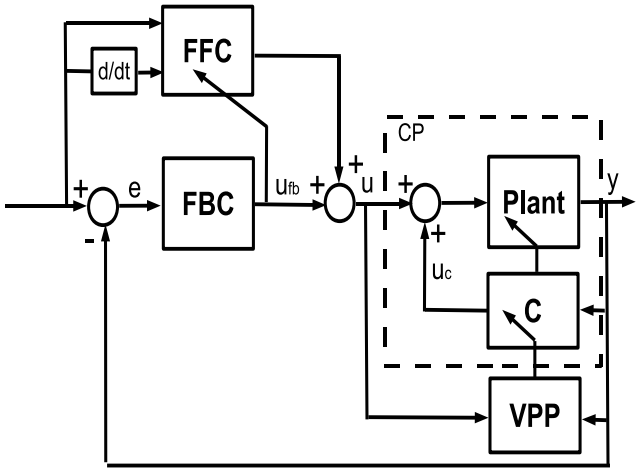

Fig. 4. VPP + FEL

that the FEL performs better with a PD controller rather than a PID controller. It was also clear that the better the original PD controller, tuned for the desired plant, then the FEL performance was improved. However, with many very different initial configurations of the PD controller, the FEL finally learned the inverse model and the feedback control action tended to zero. This work represents the first attempt to perform simultaneous and on-line (a) plant identification, (b) controller design and (c) inverse model learning. It represents an adaptive control system, and it will potentially work even if the objective plant and controller are initially unknown performing an automatic controller design scheme based on a desired plant performance.

VI. Simulations

In this section various simulations will be performed in order to show the efficiency of the presented *Brain Style* control scheme. In our simulations, we will use two different state space models. One referring to a linear plant working as the desired plant, and the other plant will be nonlinear plant as the objective plant to be controlled.

In all our simulations we will consider the objective plant is unknown, and that all the PB models of the control scheme are initialized as all-zero matrices.

The linear plant used in the simulations is represented as,

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = ax_1 + bx_2 + r \end{cases} \quad (32)$$

where $a = -0.3$ and $b = -1.1$. Its PB representation of the f_2 is the following.

TABLE I. f_2 of the Desired Linear Plant.

$x_1 \setminus x_2$	d_2= -3	d_2= -1.5	d_2= 0	d_2= 1.5	d_2= 3
d_1= -4	4.5	2.85	1.2	-0.45	-2.1
d_1= -2	3.9	2.25	0.6	-1.05	-2.7
d_1= 0	3.3	1.65	0	-1.65	-3.3
d_1= 2	2.7	1.05	-0.6	-2.25	-3.9
d_1= 4	2.1	0.45	-1.2	-2.85	-4.5

The second plant is nonlinear, and corresponds to the so

called "*Van der Pol*" oscillator. This system is expressed as,

$$\begin{cases} \dot{x_1} = x_2 \\ \dot{x_2} = -x_1 + (1 - x_1^2)x_2 + u + r \end{cases} \quad (33)$$

Its optimal PB representation is represnted in Table II.

TABLE II. f_2 OF THE *Van der Pol* SYSTEM.

$x_1 \setminus x_2$	d_2= -3	d_2= -1.5	d_2= 0	d_2= 1.5	d_2= 3
d_1= -4	47.16	25.58	4	-17.58	-39.16
d_1= -2	8.96	5.48	2	-1.48	-4.96
d_1= 0	-4.97	-2.49	0	2.49	4.97
d_1= 2	4.96	1.48	-2	-5.48	-8.96
d_1= 4	39.16	17.58	-4	-25.58	-47.16

The performance of the implemented control scheme is represented in the figures where the following parameters are shown:

(a) Desired Plant performance controlled with only the Feedback (FB) controller.

(b) Plant controlled with the Brain Style control scheme (VPP + FEL).

(c) Feedback action (u_{fb}) of the original controller within the FEL scheme.

(d) Feedforward action (u_{ff}) of the feedforward controller as it is updated and the total control action (u).

In (a) and (b) the blue line represents the reference input (r) for the plant and the green line corresponds to the output of the plant (y).

A. Simulation 1

In the first simulation shown in Fig. 5, the feedback controller was chosen to be a PD controller with gains $K_P = 5$ and $K_D = 1$.

Fig. 5 shows the performance of the *Brain Style* control scheme for a sinusoidal input reference. The feedback action of the PD controller tends to zero while the feedforward controller is learned. The performace of the objective plant is similar to the desired plant, and the control performance is better due to the feedforward action of the FEL.

In Fig. 6 it can be observed how the performance of the objective plant in the *Brain Style* control scheme (in red), becomes similar, while C is learned, to the performance of the desired plant (DP). The black line show how the objective plant would perform only by using a feedback controller.

The estimated objective plant value in this case is shown in Table III. The estimated values are very close to the values shown in Table II, apart from those vertexes which did not have enough data to be estimated correctly.

TABLE III. ESTIMATION OF OBJECTIVE PLANT.

$x_1 \setminus x_2$	d_2= -3	d_2= -1.5	d_2= 0	d_2= 1.5	d_2= 3
d_1= -4	0	24.2650	4.4304	-15.7054	0
d_1= -2	10.7306	5.5153	2.3574	-1.5226	2.2520
d_1= 0	-13.1594	-2.4861	0	2.5059	7.3804
d_1= 2	-1.7338	1.5087	-2.1238	-5.5592	0
d_1= 4	0	15.2068	-4.2249	-23.4683	0

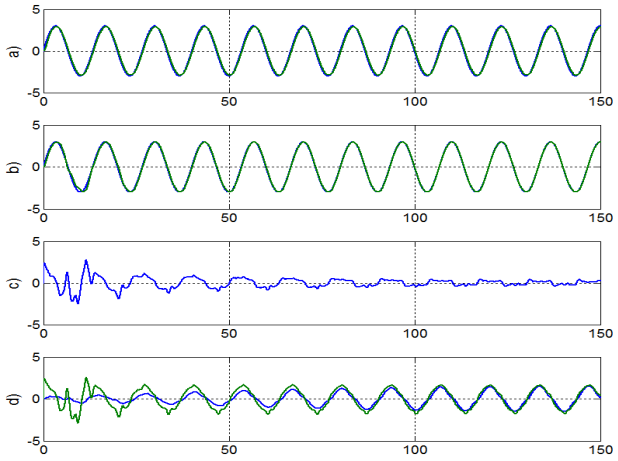

Fig. 5. (a) FB scheme of desired plant, (b) Brain Style control scheme on the objective plant, (c) u_{fb} in FEL, (d) u_{ff} & u in FEL. Feedback action of the PD controller tends to zero while the feedforward controller is learned.

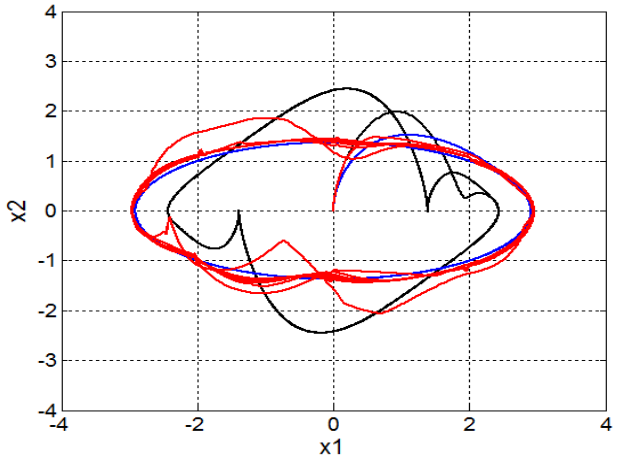

Fig. 6. Trajectories of (a) Objective plant controlled in feedbcak with a PD, in black. (b) In blue the desired plant DP controlled with a PD. (c) The objective plant in a Brain Style control scheme in red (closed loop plant CP, during the design of C and learning of inverse model.

TABLE IV. DESIGNED CONTROLLER

$x_1 \setminus x_2$	d_2= -3	d_2= -1.5	d_2= 0	d_2= 1.5	d_2= 3
d_1= -4	0	-21.2346	-3.1855	15.3046	0
d_1= -2	2.0457	-3.2845	-1.7010	0.4482	-3.7387
d_1= 0	8.5878	4.0617	0.0000	-4.1386	-8.4127
d_1= 2	2.5327	-0.4705	1.5245	3.2636	0
d_1= 4	0	-14.7877	2.9874	19.5681	0

Table IV is the controller designed on the VPP to make the objective plant P behave like the desired plant DP.

Finally, Table V shows the inverse model of the closed-loop plant learned during the simulation through FEL.

TABLE V. PSEUDO-INVERSE MODEL OF CP PLANT.

$x_1 \setminus x_2$	d_2= -3	d_2= -1.5	d_2= 0	d_2= 1.5	d_2= 3
d_1= -4	0	-0.2761	-0.1559	0.1731	0
d_1= -2	0	-1.5493	0.3818	1.2999	0
d_1= 0	0	-1.6247	0	1.4969	0
d_1= 2	0	-1.2258	-0.1624	1.4352	0
d_1= 4	0	-0.1447	0.1604	0.2122	0

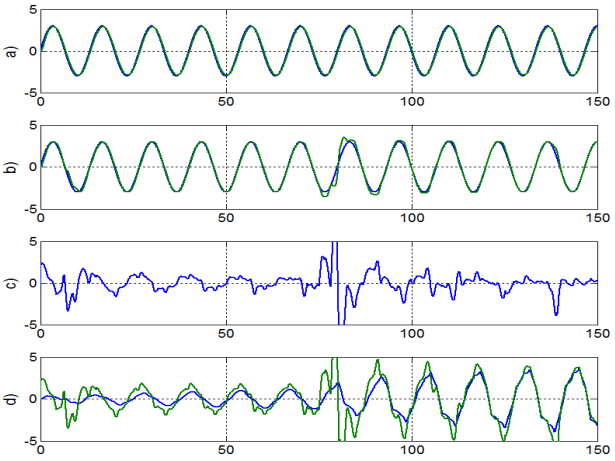

Fig. 7. Brain Style control scheme for the *Van der Pol* system. Plant characteristic changes at $time = 75seconds$ and plant must be updated.

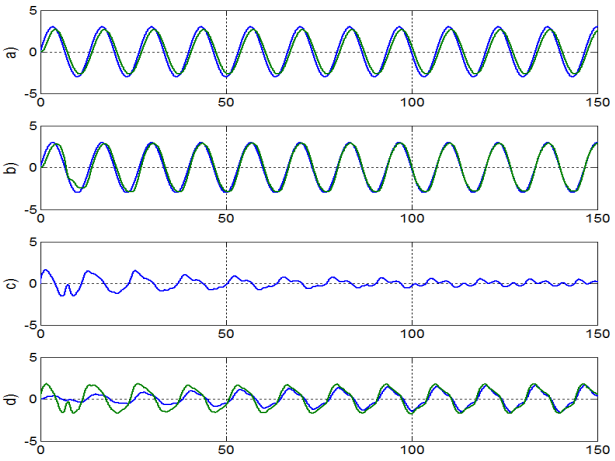

Fig. 8. (a) FB scheme of desired plant, (b) Brain Style control scheme on the objective plant, (c) u_{fb} in FEL, (d) u_{ff} & u in FEL. Feedback action of the PD controller tends to zero while the feedforward controller is learned.

B. Simulation 2

For this simulation the parameters of the Van der Pol system have been changed at $time = 75seconds$. To be precise, the change refers to $\dot{x_2}$ which is altered as $\dot{x_2} = -3x_1 + (1 - x_1^2)x_2 + u + r$ As a result, the control scheme responds as expected, where both objective plant estimation and controller C are updated and the feedback controller starts working again until the feedforward controller is updated.

C. Simulation 3

In this final simulation the feedback controller parameters have been modified to show that even with a different feedback action the FEL keeps working well and the inverse model is learned correctly. Once again a PD feedback controller is used but now with gains $K_P = 1.5$ and $K_D = 0.2$.

Fig. 8.a shows how the desired plant cannot follow perfectly the reference input using this new controller. However, in Fig. 8.b we can see that the inverse model is learned correctly through FEL and the feedback action tends to zero.

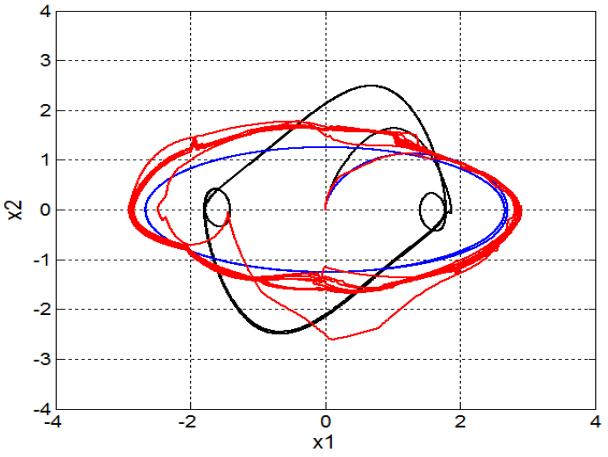

Fig. 9. Trajectories of (a) Objective plant controlled in feedbcak with a PD, in black. (b) In blue the desired plant DP controlled with a PD. (c) The objective plant in a Brain Style control scheme in red (closed loop plant CP, during the design of C and learning of inverse model.

In Fig. 9 it can be observed how blue and red lines are not similar, and that is due to the new parameters of the feedback PD controller. Table VI shows the inverse model of the closed-loop plant learned during the simulation through FEL.

TABLE VI. PSEUDO-INVERSE MODEL OF CP PLANT.

$x_1 \setminus x_2$	d_2= -3	d_2= -1.5	d_2= 0	d_2= 1.5	d_2= 3
d_1= -4	0	-0.3286	-0.7002	0.1239	0
d_1= -2	0	-1.7728	0.0775	2.2057	0
d_1= 0	0	-2.3281	0	2.4299	0
d_1= 2	0	-2.0019	0.0441	1.8151	0
d_1= 4	0	-0.1087	0.5890	0.2942	0

VII. CONCLUSIONS

In this paper we have presented the *Brain Style* control scheme. It has been inspired by the brain's ability to implement the most efficient and robust control systems available. Even though only a preliminary test of the control scheme has been presented, where there are many improvement options, it represents the first attempt to perform simultaneous and on-line (a) forward model identification, (b) controller design and (c) inverse model learning. This paper also contributes in our attempt to demonstrate the full potential of PB models for nonlinear control systems. The presented control scheme, is tested in simple situations, however it works and it shows its potential as an adaptive control scheme. If plant and controller are initially unknown the presented control scheme will perform an automatic controller design.

There are many options to explore further the *Brain Style* control scheme. In future works the effects of different feedback controllers on the FEL performance should be analyzed. Optimum vertex number and placement for the PB models and systems with time delays need to be analyzed as well. Finally, the learning sequence (gains, rates, etc.) should also be analyzed in future works, for the most efficient implementation of the presented control scheme. Depending on the dynamics of the analyzed system this could be important. For example, it could be that the FEL should initially have a very slow learning rate, until the plant identification and

controller design was implemented correctly. Another option is to start performing the whole scheme simultaneously but with very small gains in the feedback controller and increase them slowly, in order to guarantee the system's stability.

ACKNOWLEDGMENT

This work was supported in part by the Spanish Ministry of Science and Innovation (grant TIN2011-29827-C02-01). This work was also supported by a URP grant from Ford Motor Company.

REFERENCES

[1] K. Doya, H. Kimura, and M. Kawato, "Neural mechanisms of learning and control," *Control Systems, IEEE*, vol. 21, no. 4, pp. 42–54, 2001.

[2] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural Networks*, vol. 11, no. 7, pp. 1317–1329, 1998.

[3] M. Kawato, "Internal models for motor control and trajectory planning," *Current opinion in neurobiology*, vol. 9, no. 6, pp. 718–727, 1999.

[4] M. Kawato, K. Furukawa, and R. Suzuki, "A hierarchical neural-network model for control and learning of voluntary movement," *Biological cybernetics*, vol. 57, no. 3, pp. 169–185, 1987.

[5] H. Khalil, *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002, vol. 122.

[6] A. Isidori, *Nonlinear control systems*. Springer Verlag, 1995, vol. 1.

[7] E. Sontag, "Nonlinear regulation: The piecewise linear approach," *Automatic Control, IEEE Transactions on*, vol. 26, no. 2, pp. 346–358, 1981.

[8] M. Johansson and A. Rantzer, "Computation of piecewise quadratic lyapunov functions for hybrid systems," *Automatic Control, IEEE Transactions on*, vol. 43, no. 4, pp. 555–559, 1998.

[9] J. Imura and A. Van Der Schaft, "Characterization of well-posedness of piecewise-linear systems," *Automatic Control, IEEE Transactions on*, vol. 45, no. 9, pp. 1600–1619, 2000.

[10] G. Feng, G. Lu, and S. Zhou, "An approach to h? controller synthesis of piecewise linear systems," *Communications in information and systems*, vol. 2, no. 3, pp. 245–254, 2002.

[11] T. Takagi and M. Sugeno, "Fuzzy identification of system and its applications to modelling and control," *IEEE Trans. Syst., Man, and Cyber*, vol. 1, p. 5, 1985.

[12] K. Tanaka and H. Wang, *Fuzzy control systems design and analysis: a linear matrix inequality approach*. Wiley-Interscience, 2001.

[13] M. Sugeno, "On stability of fuzzy systems expressed by fuzzy rules with singleton consequents," *Fuzzy Systems, IEEE Transactions on*, vol. 7, no. 2, pp. 201–224, 1999.

[14] T. Taniguchi, L. Eciolaza, and M. Sugeno, "Nonlinear control for multiple-input and multiple-output nonlinear systems with pb models based on i/o linearization," in *The 2nd World Conference on Soft Computing,*, 2012, pp. 288–293.

[15] L. Eciolaza and M. Sugeno, "Optimal piecewise bilinear modeling of nonlinear systems," in *Advances on Computational Intelligence*. Springer, 2012, pp. 91–100.

[16] ——, "On-line design of lut controllers based on desired closed loop plant: Vertex placement principle," in *IEEE International Conference on Fuzzy Systems (FuzzIEEE)*. IEEE, 2012, pp. 1–8.

[17] L. Eciolaza, T. Taniguchi, M. Sugeno, D. Filev, and Y. Wang, "Piecewise bilinear models for feedback error learning: On-line feedforward controller design," in *IEEE International Conference on Fuzzy Systems (FuzzIEEE)*. IEEE, 2013.

[18] T. Taniguchi, L. Eciolaza, and M. Sugeno, "Look-up-table controller design for nonlinear servo systems with piecewise bilinear models," in *IEEE International Conference on Fuzzy Systems (FuzzIEEE)*. IEEE, 2013.

[19] T. Taniguchi and M. Sugeno, "Stabilization of nonlinear systems with piecewise bilinear models derived from fuzzy if-then rules with singletons," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*. IEEE, pp. 1–6.

[20] M. Sugeno and T. Taniguchi, "On improvement of stability conditions for continuous mamdani-like fuzzy systems," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 120–131, 2004.