Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

PB model-based FEL and its application to Driving Pattern Learning

Luka Eciolaza * **Tadanari Taniguchi** ** **Michio Sugeno** *
Dimitar Filev *** **Yan Wang** *** **John Michelini** ***

* *European Centre for Soft Computing, 33600 Mieres, Asturias, Spain (e-mail: luka.eciolaza@softcomputing.es, michio.sugeno@gmail.com).*
** *Tokai University, Hiratsuka, Kanagawa, 2591292 Japan (e-mail: taniguchi@tokai-u.jp).*
*** *Ford Research and Innovation Center,2101 Village Road, Dearborn, MI 48103, USA (e-mail: dfilev@ford.com, ywang21@ford.com, jmichel1@ford.com) .*

Abstract:
A significant part of the vehicle control software is based on Look-Up-Tables (LUTs) that define the mappings from different combinations of multiple independent (input) variables to the values of one dependent (output). LUTs are used as feedforward controllers or as gain-scheduling parameters for feedback controllers. The LUT feedforward controllers can be viewed as inverse vehicle models capturing the strong nonlinearity and multimodal behaviors that can be (in many cases) formalized only by experimentally measured data under different operating conditions. The paper proposes a Feedback Error Learning (FEL) based method for adaptation of the LUT feedforward controllers in order to match the desired and actual vehicle performance. The FEL is an on-line learning strategy acquiring an inverse model of a plant through feedback control actions. In this paper we consider the driver demand LUTs as a feedforward controller defining the relationship between the accelerator pedal position, the engine speed, and corresponding brake torque and the driver as a feedback controller. The FEL scheme have been implemented through Piecewise Bilinear (PB) models which can be expressed as LUTs and are very convenient with regard to nonlinear modeling, control objective and on-line learning capability.

1. INTRODUCTION

Look-up tables (LUTs) are used in automotive engineering applications as feedforward controllers or as gain-scheduling parameters for feedback controllers. In a broad sense they represent "pseudo-equations to make up for a lack of *real* equations or perhaps to replace complicated equations with simpler ones" (Bernstein [2005]). For example, the engine control systems employ from $5,500$ to $6,000$ LUTs that contain calibration parameters or define engine control actions under different operating conditions (C. Bohn and Magnor [2006]). The most common LUTs are the two-dimensional (2D) tables that define the values of one dependent (output) variable for different combinations of two independent (input) variables. Some of the reasons for the wide acceptance of the LUTs in the automotive industry are the strong nonlinearity and multimodal behaviors that can be (in many cases) formalized only by experimentally measured data under different operating conditions. The LUT feedforward controllers can be viewed as inverse vehicle models capturing the entire vehicle operation under different operating conditions. Such control is designed to fit normal driver and environment and have to compromise performance otherwise.

The typical approach to addressing the problem of adaptation of the LUT based feedforward controllers is to use the difference between the desired and the actual vehicle performance to realize an adaptation of the LUTs so as to match the two performances. Because all the LUTs are built for nominal conditions if the vehicle operates under such conditions, the vehicle performance with inputs defined by the LUTs outputs (actuators) usually match the desired performance which is the input to the LUTs. Any difference of the performance indicates the deviation from the nominal conditions (driver and/or environment). It is then possible to consider an algorithm for the modification of the LUTs to improve performance matching the desired performance.

This conventional approach of adaptation is not always applicable to the vehicle feedforward LUT controllers because the driver is not an independent controller but a part of a human-in-the-loop system. Fig. 1 shows a general view of the driver-vehicle interaction (Q. Zhao and Filev [2013]). The drivers control actions that are communicated to the vehicle subsystems through the Human Machine Interaction (HMI), steering (S), accelerator (A), and brake (B) pedals are a result of a complex process driven by two main information flows - a cognitive flow including the quantitative information readily available to the driver through various sensors, control and advisory systems (solid lines) and a subjective flow including visual, perceptual, emotional, and experiential factors that are processed in the driver's mind (broken line). While the quantitative information flow is available to both the driver and the vehicle systems, the subjective information flow is generally inaccessible to the electronic control systems. In addition, the drivers objectives (target speed, acceleration, torque, etc.) are in general not explicitly available to the vehicle control systems. Because of

⋆ This work was partially supported 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.

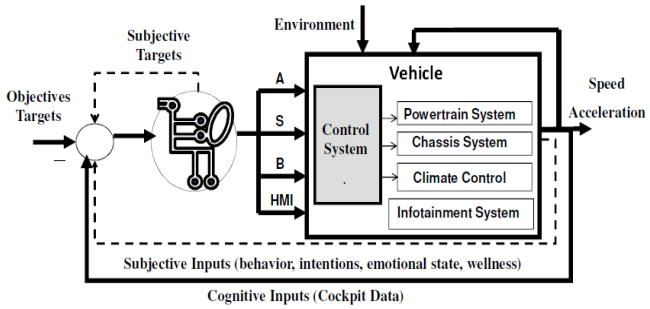

Fig. 1. Driver-Vehicle Interaction.

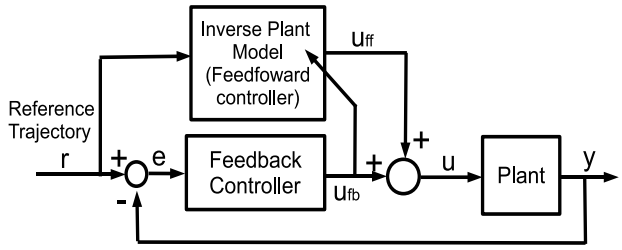

Fig. 2. Feedback error learning architecture.

these reasons it is not always easy or even possible to apply the conventional approach of developing an adaptive strategy that would adjust the feedforward LUT controllers to the actual vehicle performance.

In order to address the problem of adaptation of the vehicle feedforward LUT controllers in this paper we focus our attention on a different adaptation mechanism called the Feedback Error Learning (FEL) scheme, proposed by Kawato et al. [1987], that was originally inspired on biological control systems which usually are systems with time delays and strong nonlinearities. Fast and coordinated limb movements cannot be executed solely under feedback control, since biological feedback loops are slow and have small gains. Thus, the brain uses inverse dynamical models in order to calculate necessary feed-forward commands from desired trajectory information, see Kawato [1999]. FEL and the inverse model identification scheme represent an important role for the quick and smooth motions of the limbs in human motor control. The FEL has the feature that the learning and control can be performed simultaneously.

In most cases, human control actions are based on feedforward motions as seen in playing tennis or baseball. In fact, it is well-known that the brain has capability to acquire feedforward motions by learning through feedback motions. In particular, driver's actions for vehicle control are feedforward. Therefore, it is very effective to introduce feedforward controllers as a brain-style man-machine interface to assist driver's control actions.

A typical schematic of the FEL control system is shown in Fig. 2. It consists of the combination of a feedback controller that ensures the stability of the system and an adaptive feedforward controller that improves control performance. An inverse model as a feedforward controller is learned by FEL so as to minimize the square error between u_{ff}, the output of a feedforward controller and u_0, the ideal feedforward control based on an inverse model. Using a conventional steepest descent method for the minimization, the parameters of a feedforward controller can

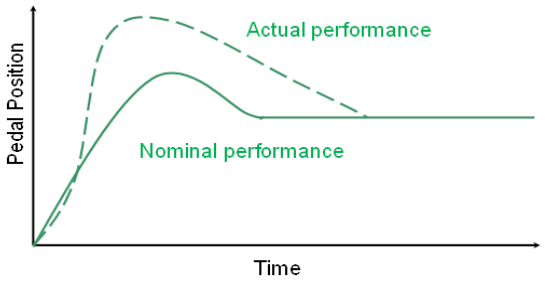

Fig. 3. Nominal vs. actual performance: (a) Aggressive vs. typical driver, or (b) same LUT controller in different vehicle conditions.

be sequentially updated in proportion to $(u_{ff} - u_0)$. However since the ideal u_0 is unknown, this error signal is approximated with u_{fb} called feedback error in FEL. As it is shown in Fig. 2, u (the control input to a plant) is equal to $u_{ff} + u_{fb}$, and $u_{ff} - u_0$ is replaced with $u_{ff} - u = u_{fb}$. After learning is complete, i.e., $y(t) = r(t)$, then u_{fb} tends to zero and feedback control is replaced by feedforward control. Thus, $u = u_{ff}$ the feedforward controller should be serving as the inverse of the original plant.

In this paper, we have taken the approach to implement FEL based on Piecewise Bilinear (PB) models recently proposed in Eciolaza et al. [2013]. In this work we concluded the PB model represents the most convenient model to implement FEL with regard to nonlinear modeling, control objective and on-line learning capability.

FEL scheme has been successfully applied to a variety of control tasks in the past, as in Jung and Kim [2008], Ruan and Chen [2011], Nakamura et al. [2011]. However the inverse model identification was normally implemented via neural networks (NN), which are not convenient for control purpose. On the other hand, the successes of FEL attracted attention in control system community, so Miyamura and Kimura [2002], Miyamura Ideta [2006] and AlAli et al. [2006], proved FEL's stability for SISO linear systems with a null relative degree, systems with time delay and MIMO plants respectively. However, these analysis were only made for linear systems.

The PB model, proposed in Sugeno [1999], is very effective for modeling and control of nonlinear systems. It is a fully parametric model to represent Linear/Nonlinear systems. PB model has a big number of parameters to tune, but Eciolaza and Sugeno [2012] showed that the global tuning of PB models can be made in a very simple and efficient way. PB models are very convenient for control purpose as they are interpretable (for instance as LUT), easy to compute and simple to handle.

An example of the potential of using FEL in automotive control is for learning the driver pattern during vehicle launch or tip-in, and adjusting vehicle control look-up tables to match the driver's demand. Fig. 3 shows the pedal profiles of an aggressive driver, dashed line named *Actual performance*, compared to a typical driver, line named *Nominal performance*, during vehicle launch.

In this case, the vehicle controller (LUT) serves a feedforward controller that is the inverted vehicle plant and delivers control signals based on driver demands. The initial LUT is calibrated for a normal driver, with a nominal pedal profile. Therefore, the controller does not deliver the desired vehicle acceleration

for an aggressive driver with a different pedal profile. It is then desirable to have an adaptive mechanism to learn the driver's behavior and adjust the controller to match his/her behavior. The difference between the nominal and actual pedal profile will prompt the correction of the LUTs to deliver the response the driver wants.

The algorithm adapts the fixed-parameter vehicle control system to maintain desired performance when the vehicle is driven at varying conditions such as, different driver, different environment, and different vehicle conditions. Such condition changes will make vehicle performance deviating from nominal.

Sections II introduces the PB model-based FEL scheme, where both PB models and FEL architecture are explained. Section III describes the vehicle control application and the implementation of FEL within it. Section IV presents experimental results. Section VI concludes the paper with some discussion and final remarks of the results.

2. PB MODEL-BASED FEL

As is explained in the previous section, Fig. 2 illustrates the feedback error learning architecture. It consists of an objective plant to be controlled with a feedback controller and, in addition, a feedforward controller to be learned by FEL. The objective of control is to minimize the error e between the reference signal r and the plant output y.

If the mapping from the motor command u to the sensory outcome y is given by $p : u \rightarrow y$, for a given reference input r , the ideal control command, which makes the tracking perfect, is given by the inverse $u(t) = p^{-1}(r(t))$. However, the direct inverse modeling, using I/O data, does not necessarily guarantee achieving a particular target trajectory even when the training period is sufficiently long Kawato [1990].

In the FEL approach, the feedback controller action is converted into motor command error and used to learn the feedforward controller. By FEL, eventually the feedback control is replaced by feedforward control. The learning algorithm used in FEL is the same independent of models: NN model, linear model and our PB model. FEL includes the features of simultaneous learning and control, making it an adaptive controller.

2.1 PB models

The PB model is a fully parametric model to represent Linear/Nonlinear systems. It is designed to be easily applicable for control purpose. 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 PB model can be expressed as a LUT which is widely used to realize controllers in industrial applications, due to its interpretability, visibility and simplicity. A local error does not trigger a global error and its interpolation nature generates robust outputs.

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

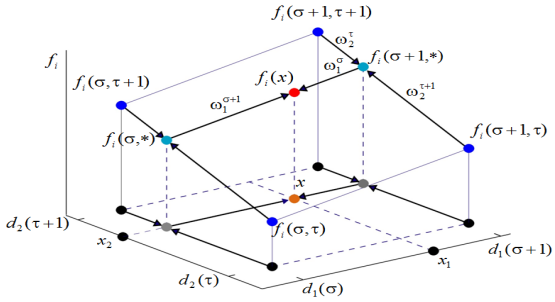

Fig. 4. Piecewise region $R_{\sigma\tau}$ and interpolation of $f_i(x)$.

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

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 Sugeno [1999] 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 $(1 \leq \sigma \leq n_1, 1 \leq \tau \leq n_2)$ defined by,

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

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

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. 4. Note that the approximation is made by only using the values of a nonlinear function at the vertexes of R_{ij}'s in (3).

2.2 FEL: On-line Sequential Learning Algorithm

Let us express a plant model for the sake of simplicity 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 $u_{ff} = p^{-1}(r)$. We consider a feedforward controller expressed as $u_{ff} = p^{-1}(r, \dot{r})$, where r is a desired output y_d and $\dot{r}$ is $\dot{y}_d$.

In what follows, we deal with a feedforward controller with two inputs r, $\dot{r}$ and single output u_{ff}. This feedforward controller in the FEL scheme is interpreted as a pseudo-inverse model of an objective plant. Note that a plant is, in general, not invertible unless its relative degree is 0. We will initially assume that an objective plant is unknown and its feedback controller is given. In a realistic nonlinear system control scenario, both plant identification and controller design could also be performed through PB models.

Let u_0 be an ideal feedforward control based on a pseudo-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

$$I = \frac{(u_{ff} - u_0)^2}{2} \tag{11}$$

where the PB representation of the feedforward controller is,

$$u_{ff} = p^{-1}(r, \dot{r}) = \sum_{i=\sigma}^{\sigma+1} \sum_{j=\tau}^{\tau+1} w_1^i(r) w_2^j(\dot{r}) V(i,j) \tag{12}$$

I can be sequentially minimized using the derivative of (11)

$$\frac{\partial I}{\partial V} = \frac{\partial u_{ff}}{\partial V}(u_{ff} - u_0). \tag{13}$$

However, the error $(u_{ff} - u_0)$ is not available since u_0 is unknown. Therefore Kawato Kawato et al. [1987] 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

$$\frac{\partial I}{\partial V} = \frac{\partial u_{ff}}{\partial V} u_{fb} = \frac{\partial p^{-1}(r, \dot{r})}{\partial V} u_{fb} \tag{14}$$

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

$$V_{new(i,j)} = V_{old(i,j)} - \delta \frac{\partial u_{ff}}{\partial V(i,j)} u_{fb} \tag{15}$$

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 (12), with (8) (9) and (10), we have:

$$\begin{aligned} u_{ff} = (1-\alpha)(1-\beta)V_{(\sigma,\tau)} + (1-\alpha)\beta V_{(\sigma,\tau+1)} \\ +\alpha(1-\beta)V_{(\sigma+1,\tau)} + \alpha\beta V_{(\sigma+1,\tau+1)} \end{aligned} \tag{16}$$

(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 equations of change for each vertex of a region will be:

$$\begin{cases} V_{new(\sigma,\tau)} &= V_{old(\sigma,\tau)} - \delta(1-\alpha)(1-\beta)u_{fb}, \\ V_{new(\sigma,\tau+1)} &= V_{old(\sigma,\tau+1)} - \delta(1-\alpha)\beta u_{fb}, \\ V_{new(\sigma+1,\tau)} &= V_{old(\sigma+1,\tau)} - \delta\alpha(1-\beta)u_{fb}, \\ V_{new(\sigma+1,\tau+1)} &= V_{old(\sigma+1,\tau+1)} - \delta\alpha\beta u_{fb} \end{cases} \tag{17}$$

How to choose the value of δ will be discussed in the experimentation section.

3. DRIVER PATTERN LEARNING

In this paper we will demonstrate the proposed FEL scheme for driving pattern learning. For that, a simulink model for vehicle speed control has been used. The model simulates the performance of a car for various given speed profiles (i.e.: city profile, highway profile).

The model has a controller implemented in order to make the vehicle follow the target speed at each point. The feedback action of the controller is divided in both pedal and brake actions, where pedal and brake actions are always positive and within [0,100].

The full speed profile (V_{target}) is given for the simulations and at every sample the speed error $(V_{target} - V_{actual})$ is known and the controller tends to minimize the error.

3.1 FEL for Driver Pattern recognition:

In this application of the vehicle speed control system, the implementation of the FEL algorithm changes with respect to Eciolaza et al. [2013]. In this case, the FEL diagram is designed so that both *Pedal* and *Brake* actions are learned. For that, an unification of positive pedal input and negative brake input is done. The input u to the car takes both positive and negative values where positive value is realized by pedal and negative by brake, mechanically. If u_{ff} is positive then activate pedal and if negative then activate brake. Therefore we use pedal (u_{fb_p}) and brake (u_{fb_b}) actions, which are unified as $u_{fb} = u_{fb_p} - u_{fb_b}$ theoretically.

Pedal-off is the engine brake and so there is no theoretical border between pedal-on, pedal-off and braking. In such a way, pedal action and brake action are continuous and we have to reflect this fact in FEL for both learning and feedforward controller. Fig. 5 shows the diagram of how FEL has been implemented in the application.

The two inputs to the car, pedal and brake, control the acceleration, positive or negative, of the car:

- Pushing pedal increases the acceleration: $++$

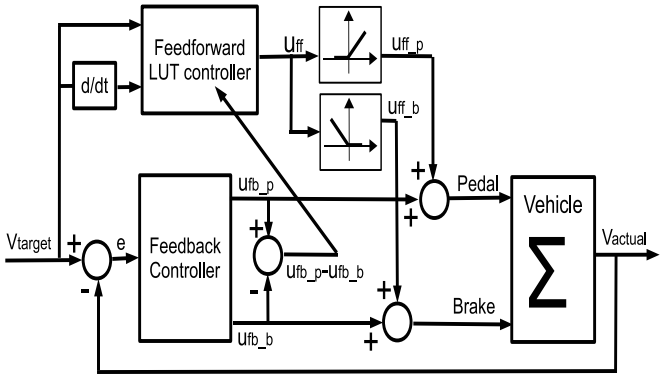

Fig. 5. FEL scheme implemented for driving pattern learning.

- Releasing (weakening) pedal relatively decreases the acceleration: $+$
- Pedal-off is the so-called engine brake to decrease acceleration: $-$
- Brake is to make pedal-off more effective by creating negative input: $--$

So the vehicle input u for acceleration is considered as

$$u = Pedal - Brake, \tag{18}$$

where either pedal or brake is used at each time instant and pedal means positive u and brake means negative u.

Along this consideration, we should formulate as

$$Pedal = u_{fb_p} + u_{ff_p}, \tag{19}$$

$$Brake = u_{fb_b} + u_{ff_b}, \tag{20}$$

$$u_{fb} = u_{fb_p} - u_{fb_b} \tag{21}$$

Both u_{fb_p} and u_{fb_b} are used for FEL but u_{fb_b} does not contribute in (19), and u_{fb_p} does not contribute in (19). And

$$u_{ff_p} = \begin{cases} 0 & ,if\ u_{ff} < 0 \\ u_{ff} & ,if\ u_{ff} > 0 \end{cases} \tag{22}$$

$$u_{ff_b} = \begin{cases} -u_{ff} & ,if\ u_{ff} < 0 \\ 0 & ,if\ u_{ff} > 0 \end{cases} \tag{23}$$

4. EXPERIMENTATION

The implementation of the FEL scheme in the application has been done successfully and learned feedforward controller representing the pseudo-inverse of the plant is stable. By FEL, feedback control is replaced by feedforward control. The u_{fb_p} and u_{fb_b} actions are minimized and when learning is complete they tend to zero.

Between the different parameters that must be tuned, the adequate number of vertexes and regions must be defined. In this case these values have been selected depending on different speed profiles available for the simulations.

On the other hand, the update rate (K_{updt}) of the feedforward controller must be defined. This value should depend on the sample rate of the system and the number of vertexes of the system. In this case, as the sample rate (sr) has been variable in our simulation set-up, we chose $K_{updt} = 0.01 \times sr$.

Finally, as the analyzed system has a time-delay, we have used a pure time lag of $\tau = 1.4$ seconds for the implementation of the FEL scheme. Take into account that the delayed signal is only

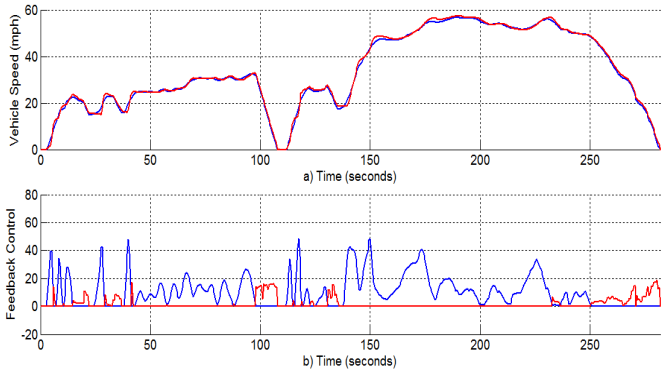

Fig. 6. Simulation of maneuver with no FEL.

used for the learning of the feedforward controller. Correction action $u_{fb}(t)$ corresponds to input references $(r(t-\tau), r_dot(t-\tau))$. The feedforward controller will anticipate the time delay (τ) of the system.

Note that the origin ($V_{target} = 0, ACC_{target} = 0$) vertex has been set to zero.

The performance of the implemented FEL scheme is represented in figures (Fig. 6, Fig. 7, Fig. 8) where the following parameters are shown:

(a) Vehicle Speed where the desired speed (V_{target}) in blue and the real vehicle speed (V_actual) in red.
(b) Pedal feedback action (u_{fb_p}) in blue and the Brake feedback action (u_{fb_b}) in red.
(c) Feedforward controller action (u_{ff}) in blue, the total Pedal action ($Pedal = u_{fb_p} + u_{ff_p}$) in green and the total Brake action ($Brake = u_{fb_b} + u_{ff_b}$) in red.

Fig. 6 shows an example of two consecutive acceleration maneuvers of the vehicle with no FEL scheme implementation. When V_{actual} does not meet the V_{target}, then pedal is pushed to accelerate the car. The car has its inertia and pedal is released when car is accelerating. The error between desired and actual vehicle speeds is not big, and the total pedal and brake actions are determined by the feedback controller.

Fig. 7 shows an example of the same two consecutive accelerations maneuvers of the vehicle working with the FEL scheme. The feedfoward model is initialized as a null matrix and it can be observed, while learning, how the total pedal and brake actions eventually are shifting to be determined by the feedforward controller.

Fig. 8 shows the vehicle working with the FEL scheme. In this case the feedfoward model is already learned. The total pedal and brake actions are mainly determined by the feedforward controller, and the feedback actions are minimized.

Table I shows the feedforward controller learned in the simulation presented in Fig. 8.

5. CONCLUSION

In this paper we have investigated the application of PB model based FEL for intelligent vehicle systems environment. A method to make an on-line sequential learning of a feedforward controller as a PB model has been used. The FEL scheme controls a system by adapting the pseudo-inverse piecewise bilinear model based on an output of a feedback controller, and

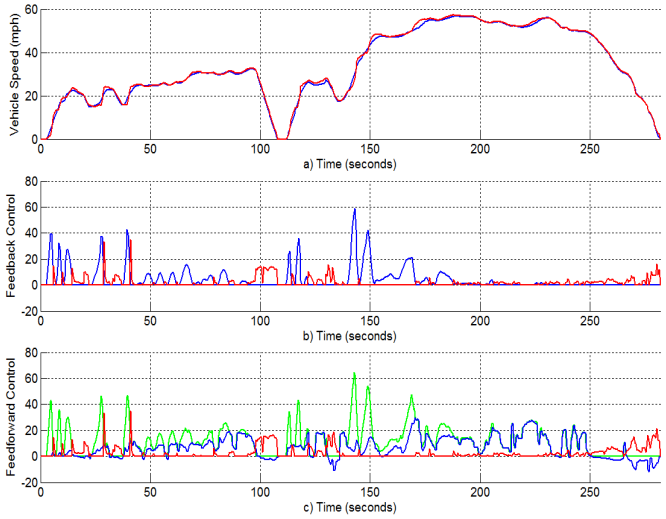

Fig. 7. Simulation of maneuver with FEL. Initialized to null and while learning.

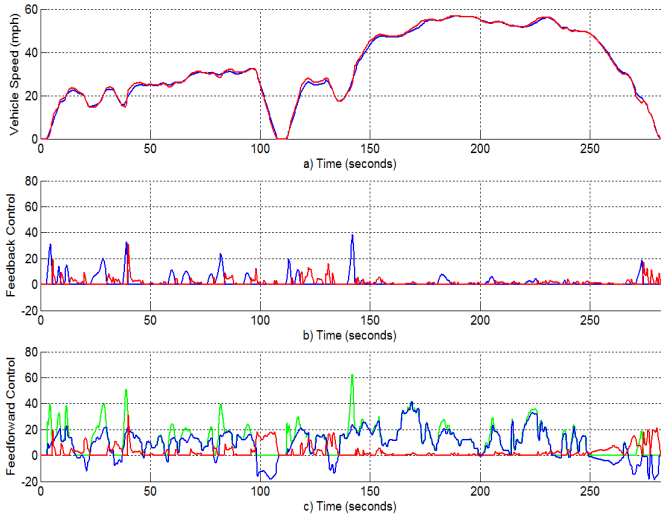

Fig. 8. Simulation of maneuver with FEL. Inverse model is learned already.

Table 1. Learned feedforward controller.

$\dot{y}_d \setminus y_d$	d_1=0	d_1=10	20	30	40	50	60
d_2= -4	-6.3	-14.6	-10.2	-2.4	0	0	0
d_2= -3	-18.7	-19.4	-13.3	-15.7	-0.1	0	0
d_2= -2	-0.2	1.4	-2.9	-9.8	-3.9	-0.3	0.1
d_2= -1	-0.1	1.2	-6.4	-7	-6	-2.3	-0.7
d_2= -0.5	-0.01	4.1	-5.6	-1.5	0.3	-1.8	1.3
d_2= 0	0	14.4	4.7	8.5	8.8	13.6	8.1
d_2= 0.5	13.4	16.2	11.1	15.7	7.5	44.6	9.1
d_2= 1	26.8	11.5	11.9	11.4	26	10.6	3.3
d_2= 2	18.6	2.5	19.1	25.7	25	1.6	0
d_2= 3	0.9	17.7	22.2	23.2	1.9	0	0
d_2= 4	-8.8	1.3	5.9	0.01	-0.1	0	0

adjusting operation of an actuator in response to the learned pseudo-inverse model. The lookup table provides a near exact inverse model of the plant under control when output of the feedback controller reaches zero. Consequently, an actuator or plant command output from the lookup table may cause the plant to follow a desired trajectory that was a basis for indexing data in the lookup table.

In particular, the proposed FEL scheme has been used for driving pattern learning, where learning of both the *Pedal* and *Brake* actions has been the main goal. The FEL implementation has been done successfully, and once the learning is completed the feedback control is minimized and partially replaced by the feedforward controller.

In future works, the proposed FEL method will potentially be used to address some automotive control applications. On the one hand, the target will be to improve the control performance of traditional feedback loops. On the other hand, the adaptation of the LUTs controlling the car will be made, in order to match the desired vehicle performance and the real vehicle performance.

REFERENCES

B. AlAli, K. Hirata, and K. Sugimoto. Generalization of feedback error learning (fel) to strictly proper mimo systems. In *American Control Conference, 2006*, IEEE, 2006.

D. Bernstein. From the editor - theorem and look-up table. *IEEE Control Systems*, 24(6):6, 2005.

P. Stober C. Bohn and O. Magnor. An optimization-based approach for the calibration of lookup tables in electronic engine control. *IEEE Conference on Computer Aided Control Systems Design*, pages 2316–2320. IEEE, 2006.

L. Eciolaza and M. Sugeno. On-line design of lut controllers based on desired closed loop plant: Vertex placement principle. In *IEEE International Conference on Fuzzy Systems (FuzzIEEE)*, pages 1–8. IEEE, 2012.

L. Eciolaza, T. Taniguchi, M. Sugeno, D. Filev, and Y. Wang. Piecewise bilinear models for feedback error learning: Online feedforward controller design. In *IEEE International Conference on Fuzzy Systems (FuzzIEEE)*. IEEE, 2013.

S. Jung and S.S. Kim. Control experiment of a wheel-driven mobile inverted pendulum using neural network. *Control Systems Technology, IEEE Trans.*, 16(2):297–303, 2008.

M. Kawato. Feedback-error-learning neural network for supervised motor learning. *Advanced neural computers*, 6(3):365–372, 1990.

M. Kawato. Internal models for motor control and trajectory planning. *Current opinion in neurobiology*, 9(6):718–727, 1999.

M. Kawato, K. Furukawa, and R. Suzuki. A hierarchical neural-network model for control and learning of voluntary movement. *Biological cybernetics*, 57(3):169–185, 1987.

A. Miyamura and H. Kimura. Stability of feedback error learning scheme. *Systems & control letters*, 45(4):303–316, 2002.

A. Miyamura Ideta. Stability of feedback error learning method with time delay. *Neurocomputing*, 69(13):1645–1654, 2006.

Y. Nakamura, K. Morimoto, and S. Wakui. Experimental validation of control for a positioning stage by feedback error learning. In *Advanced Mechatronic Systems (ICAMechS), 2011 International Conference on*, pages 11–16. IEEE, 2011.

J. Brine Q. Zhao and D. Filev. Cybernetics: where shall we go? In *Proc. of the 2013 IEEE Int. Conference on Cybernetics*, pages 25–31. IEEE, 2013.

X. Ruan and J. Chen. On-line nnac for a balancing two-wheeled robot using feedback-error-learning on the neurophysiological mechanism. *Journal of Computers*, 6(3):489–496, 2011.

M. Sugeno. On stability of fuzzy systems expressed by fuzzy rules with singleton consequents. *Fuzzy Systems, IEEE Transactions on*, 7(2):201–224, 1999.