Hierarchical Fuzzy Controllers for Explicit MPC Control Laws: Adaptive Cruise Control Example

Andrés Gersnoviez, María Brox, and Iluminada Baturone

***Abstract*— This paper presents a methodology to approximate explicit Model Predictive Control (MPC) laws by hierarchical fuzzy systems, particularly piecewise-affine hierarchical (PWAH) systems. These hierarchical controllers provide high operation speed with low cost in computational and memory resources because they are formed only by single-input and single-output (SISO) fuzzy modules connected in cascade. The methodology employs the CAD tools of Xfuzzy environment to describe, adjust, verify, and implement the controllers in a Field Programmable Gate Array (FPGA). The methodology is illustrated with the design and FPGA implementation of a hierarchical controller for a car adaptive cruise control (ACC) system. The resulting controller is better in terms of speed and FPGA resource consumption than other solutions reported in the literature.**

Keywords—hierarchical fuzzy systems, fuzzy control, fuzzy CAD tools

I. Introduction

Model Predictive Control (MPC) obtains the control action by solving a finite horizon open-loop optimal control problem at each sampling instant. The high computational cost involved to solve the optimization problem online is an obstacle to a wider use of MPC controllers. This has limited their use to applications of relatively slow dynamics. A solution proposed to reduce this drawback is the use of explicit MPC controllers, which solve the optimization problem offline and compute the optimal control action, $u(x) \in \mathbb{R}^m$, as an "explicit" function of the state variables, $x(t) \in \mathbb{R}^n$, within a given domain, D, which is assumed to be polytopic. Hence, online optimization is reduced to a simple function evaluation, $u(x): D \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ [1][2]. In most cases, as explained in [2], such a function is piecewise affine (PWA) in the state variables, as follows:

$$u(x) = F_i x + g_i \quad if \quad x \in \mathcal{P}_i, \qquad i = 1, \dots, I \tag{1}$$

Where $F_i \in \mathbb{R}^{m \times n}$, $g_i \in \mathbb{R}^m$ (i = 1,…, I), and $\mathcal{P}_i \subset D$ are I non overlapped regions ($\mathcal{P}_i \cap \mathcal{P}_j = \emptyset$ for $i \neq j$), called polytopes, which form a polyhedral partition of the domain D, fulfilling the relation $\cup_{i=1}^{I} \mathcal{P}_i = D$. The domain D is divided by H $n-1$ dimensional hyperplanes of the type $h_j^T x + k_j = 0$ (j = 1,…, H), with $h_j \in \mathbb{R}^n$ and $k_j \in \mathbb{R}$. Each hyperplane divides the domain into two parts and their intersection generates the edges of the I polytopes.

Several implementations of explicit MPC controllers have been proposed in the literature. Some of them implement the optimal solution without any approximation, like the so-called Generic PWA (PWAG) [3], and the Lattice PWA (PWAL) [4]. Others replace the optimal solution by a sub-optimal one, so as to gain in latency and memory requirements, like the so-called Simplicial PWA (PWAS) [5] and the Hyperrectangular PWA (PWAR) [6]. The problem is that the complexity of several sub-optimal explicit MPC implementations increases exponentially with the number of state variables.

Another implementation of explicit MPC controllers, referred to as PWAH (PWA Hierarchical), was introduced in [7]. PWAH systems approximate the optimal MPC control law and apply a methodology that decomposes it, through the use of hierarchy, in modules of minimum dimension. They achieve a high speed and require very few computational and memory resources, which increase linearly instead of exponentially with the number of state variables.

One of the problems of fuzzy controllers with many state variables is also the *curse of dimensionality* or exponential *rule explosion*. To avoid this problem, the hierarchical fuzzy controllers presented in [8] exploit the idea of dividing a single controller into several low-dimensional subsystems connected in cascade so that the growth of the number of rules become linear rather than exponential with the number of inputs.

Fuzzy systems, as discussed in [9], may have a piecewise polynomial behavior under certain conditions. In particular, they can provide a PWA behavior. This paper proposes a new type of hierarchical fuzzy controllers and shows how PWAH controllers are a particular case of them. A methodology to design them automatically with the aid of software for soft computing, particularly the Xfuzzy environment, is described.

The paper is organized as follows. Section II presents a new type of hierarchical fuzzy controllers in which single-input single-output (SISO) modules are used, and shows their relation to PWAH implementations. Section III explains the design methodology of hierarchical fuzzy controllers with an adaptive cruise control example. Section IV summarizes how the CAD tools of Xfuzzy environment facilitate the design. Finally, conclusions and future work are given in Section V.

*Research supported by the Projects AT17_5926_USE and US-1265146 (Programa Operativo FEDER 2014-2020 and Consejería de Economía, Conocimiento, Empresas y Universidad de la Junta de Andalucía), TEC2017-83557-R and RTC-2017-6595-7 (AEI/FEDER UE).

A. Gersnoviez and M. Brox are with the Department of Electronic and Computer Engineering, Universidad de Córdoba, 14071 Córdoba, Spain (phone: +34 957212224; e-mail: andresgm@uco.es; mbrox@uco.es).

I. Baturone is with the Instituto de Microelectrónica de Sevilla (IMSE-CNM), Universidad de Sevilla, CSIC, 41092 Seville, Spain (e-mail: lumi@imse-cnm.csic.es).

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

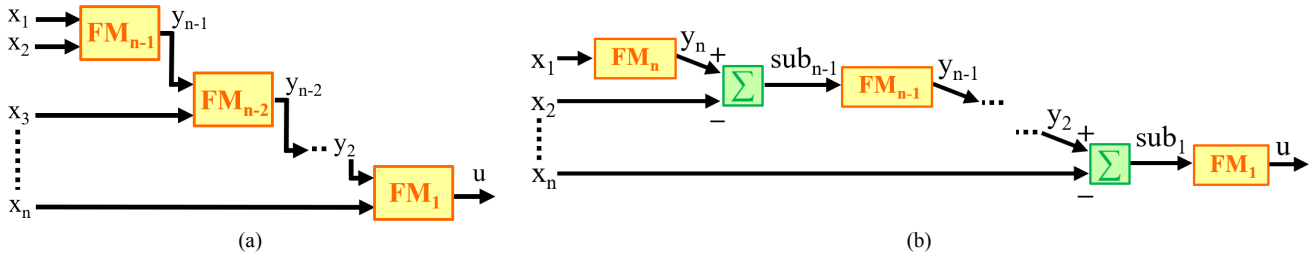

Figure 1. (a) Hierarchical decomposition proposed by Raju et al. in [8] with a minimum number of rules. (b) Hierarchical decomposition proposed herein with fewer rules.

II. Proposal of Hierarchical Fuzzy Controllers

A. Providing the smallest number of rules

Hierarchical fuzzy systems were first introduced by Raju et al. in [8]. In this paper, the authors determine that the hierarchical decomposition shown in Fig. 1(a), which uses fuzzy modules (FM) with two inputs connected in series, achieves the smallest number of rules. Specifically, this number is $(n-1)m^2$, assuming n inputs, and that the universe of discourse of all inputs is described by m fuzzy sets (for simplicity).

However, if fuzzy modules are combined with purely arithmetic modules, such as addition/subtraction, further hierarchical simplification is obtained, decreasing even more the number of rules. Specifically, the hierarchical structure we propose is shown in Fig. 1(b), where n SISO fuzzy modules (FM) are connected in cascade, so that the output of a FM is added/subtracted with another input and the result of this operation is the input of the next FM.

The proposed hierarchical fuzzy controllers meet the following theorem, which is the case illustrated in Fig. 1(b).

Theorem 1: *In a hierarchical structure with n input variables, L hierarchical modules, $m \geq 2$ (assuming m the number of fuzzy sets in the universe of discourse for all inputs), $n \geq 2$ and $n_i \geq 1$ (where n_i is the number of inputs of the i-th module), the total number of rules will reach the minimum value when $n_i = 1$ $(\forall i = 1, \ldots, L)$ and, therefore, $L = n$.*

This theorem was already demonstrated by Raju et al. in their work, but providing $(n-1)m^2$ as the minimum number of rules for the case illustrated in Fig. 1(a). However, if the universes of discourse of all inputs are described by m fuzzy sets, then each FM of Fig. 1(b) has m rules, so that the total number of rules is $n \cdot m$ in the proposed hierarchical fuzzy controller. As $n \geq 2$, the following relationship is met:

$$n + n \geq 2 + n \Leftrightarrow 2n - 2 \geq n \Leftrightarrow 2(n-1) \geq n$$

Taking the above inequality and $m \geq 2$ into account, we have:

$$n \cdot m \leq 2(n-1) \cdot m \leq (n-1) \cdot m^2 \qquad (2)$$

Therefore, the proposed hierarchical fuzzy controller shown in Fig. 1(b) uses fewer rules than the structure in Fig. 1(a).

B. The case of PWAH controllers

The proposed hierarchical fuzzy controllers meet also the following theorem

Theorem 2: *If the SISO fuzzy modules employed in the hierarchical structure are PWA, the whole hierarchical system in Fig. 1(b) will have a PWA behavior.*

Let us explain this theorem considering an explicit MPC controller with $x(t) = \{x_1, \ldots, x_n\} \in \mathbb{R}^n$ as the state vector and $u(t) \in \mathbb{R}$ as the control action. Theorem 2 states that the PWAH implementation presented in [7] is equivalent to the hierarchical structure of Fig. 1(b) when the SISO fuzzy modules employed are PWA.

A SISO PWA FM_j, with input in_j and output out_j, divides the universe of discourse of in_j into I_j non overlapped intervals and provides a linear output according to the interval i which the input belongs to:

$$out_j = f_{ji} \cdot in_j + g_{ji} \qquad (3)$$

The number of polytopes in a PWAH implementation with n SISO modules is $\prod_{j=1}^{n} I_j$. The search complexity of the polytope $\mathcal{P}_i$ where the state vector is located is logarithmic in the number of polytopes, $\mathcal{O}(n)$, because each SISO FM_j explores its intervals I_j to find the interval i associated to the polytope $\mathcal{P}_i$ (referred to as $I_j^{(i)}$). That is, if $x \in \mathcal{P}_i$, then (as shown in Fig. 1(b)), $x_1 \in I_n^{(i)}$, $sub_{n-1} = (y_n - x_2) \in I_{n-1}^{(i)}$, …, and $sub_1 = (y_2 - x_n) \in I_1^{(i)}$.

Therefore, the module which provides the control action (FM_1 in Fig. 1(b)), verifies for the polytope $\mathcal{P}_i$ that:

$$\begin{aligned} u(x) &= f_{1i} \cdot sub_1 + g_{1i} = f_{1i}(y_2 - x_n) + g_{1i} \\ &= f_{1i}y_2 - f_{1i}x_n + g_{1i} \text{ with } (y_2 - x_n) \in I_1^{(i)} \end{aligned} \qquad (4)$$

Substituting the output y_2 by $(f_{2i} \cdot sub_2 + g_{2i})$ and sub_2 by $(y_3 - x_{n-1})$, and repeating the steps until the first module in the hierarchy (FM_n in Fig. 1(b)), it can be seen that the control action provided for the polytope $\mathcal{P}_i$ is PWA in the state variables as follows:

$$\begin{aligned} u(x) = \\ f_{ni}f_{(n-1)i} \cdots f_{1i}x_1 - f_{(n-1)i}f_{(n-2)i} \cdots f_{1i}x_2 - \cdots - \\ f_{2i}f_{1i}x_{n-1} - f_{1i}x_n + g_i \text{ if } x \in \mathcal{P}_i \end{aligned} \qquad (5)$$

Where $f_{1i}, \ldots, f_{ni}$ and g_i are constants $\in \mathbb{R}$, with:

TABLE I. COMPARISON BETWEEN PWA IMPLEMENTATIONS

	Latency (number of clock cycles)	Memory (bits to store)	Multipliers	Exact optimal MPC
PWAG [3]	$\mathbf{n+2\cdot d+n\cdot d+2}$	$\mathbf{n_{bit}(n+1)(I+E)}$	1	yes
PWAL [4]	$\mathbf{n+U+1}$	$\mathbf{n_{bit}(n+1)W+U\cdot(\log_2 X+1)}$	**2n**	yes
PWAS serial [5]	$\mathbf{n+4}$	$\mathbf{n_{bit}\prod_{i=1}^{n}(I_i+1)}$	1	no (approx.)
PWAR serial [6]	$\mathbf{n+2}$	$\mathbf{n_{bit}(n+1)\prod_{i=1}^{n}(I_i+1)}$	1	no (approx.)
PWAH serial [7]	**n**	$\mathbf{n_{bit}\sum_{i=1}^{n}(3I_i-1)}$	1	no (approx.)

$$g_i = f_{(n-1)i}\cdots f_{1i}g_{ni} + f_{(n-2)i}\cdots f_{1i}g_{(n-1)i} + \cdots + f_{1i}g_{2i} + g_{1i} \text{ if } x \in \mathcal{P}_i \quad (6)$$

This equivalence is interesting since hierarchical fuzzy controllers with a PWA instead of another piecewise polynomial behavior are simpler to analyze, design, and implement, as shown in the following.

C. Advantages of PWAH Implementations

Only one clock cycle is required to process the input through a SISO PWA module and to obtain the valid input for the following SISO module in the cascade. The circuit latency (the number of clock cycles between the arrival of a new state vector value and the calculation of the corresponding control action) is, therefore, n clock cycles if n SISO modules are employed to process the n input variables.

If the number of pieces of a SISO FM$_i$ module is I_i, then the input value should be compared to $I_i - 1$ breakpoints to solve the point location problem. If the number of bits to represent the breakpoints and the parameters f_{ji} and g_i of the affine pieces is n_{bit}, the number of bits required by the PWAH implementation is:

$$n_{bit}\cdot\sum_{i=1}^{n}[(I_i-1)+2I_i] = n_{bit}\cdot\sum_{i=1}^{n}(3I_i-1) \quad (7)$$

The PWAH implementation is compared with other PWA implementations reported in the literature in Table I. The symbols employed are (besides others already defined): the depth of the binary search tree (d), the number of different local affine functions (W), the number of edges defining the polytopes (E), the number of vertices in the simplicial partition ($\prod_{i=1}^{n}(I_i+1)$), the number of rows in the simplified structure matrix of the lattice representation (X), and U, which is related to the number of super regions in the lattice representation [4]. PWAH implementations approximate the optimal MPC, like PWAS and PWAR realizations. Hence, if the optimal MPC can be approximated adequately by a PWAH solution, a high speed is provided with low cost in computational and memory resources.

III. DESIGN OF HIERARCHICAL FUZZY CONTROLLERS: ACC EXAMPLE

The next problem that arises, and which has been the subject of study until now, is to find the right hierarchical structure for a specific fuzzy system [10]-[12]. This problem is quite complex, so it is essential to follow a methodology, and use a set of CAD tools that facilitate its execution. This section focuses on hierarchical fuzzy controllers with a PWA behavior (PWAH) because of their advantages commented above. In order to explain in greater detail the methodology to design PWAH controllers presented in [7], this section describes the implementation of a problem of recent commercial interest, such as a car adaptive cruise control (ACC) system.

An ACC system is the evolution of the standard cruise control (CC) system used in most of the cars today. In this section, the ACC to be implemented is described in [13]. In this model, the speed of the *host vehicle* (v_h) and its acceleration (a_h) are available, while the relative distance between the two vehicles (x_r) and the relative velocity ($v_r = v_t - v_h$) are measured by a radar located at the *host vehicle*. The goal is to keep the *host vehicle* to a desired distance x_r from the *target vehicle* located ahead. To define this distance, it is often used the desired headway time ($t_{hw,d}$), so that $x_{r,d} = x_{r,0} + v_h \cdot t_{hw,d}$, where $x_{r,0}$ is a constant that represents the desired distance at standstill. Therefore, the tracking error is defined by $e = x_{r,d} - x_r$.

The model described in [13], along with the considerations taken in [14], is presented in the form:

$$x(t+1) = \begin{bmatrix} 1 & -T_s & 0 & Z_s \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u(t) \quad (8)$$

Where $Zs = T_s \cdot t_{hw,d} + \frac{1}{2}T_s^2$ and the state variable $x = [e\ v_r\ v_t\ a_h]^T$. The sampling time is $T_s = 0.1$s. The PWAH controller to design has three inputs because the variable v_t is considered to be constant by the model in Equation (8).

The constants mentioned above are $x_{r,0} = 3.5$m, $t_{hw,d} = 1.5$s, $v_{t,max} = 50$m/s, $v_{h,max} = 50$m/s, $a_{h,min} = -3$m/s^2, $a_{h,max} = 2$m/s^2. The constraints of the host jerk (derivative of the acceleration) are $j_{h,min} = -0.3$m/s^3 and $j_{h,min} = 0.3$m/s^3, and the radar range is $x_{rr} = 200$m.

The optimal explicit PWA controller is obtained with the following horizons (N_y, N_u) and weight matrices (Q,R):

$$N_y = N_u = 4;\ Q = \begin{bmatrix} 2.5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix};\ R = 1 \quad (9)$$

The first design step is to obtain the set of points $\mathbb{X} = \{e^{q1}, v_r^{q2}, a_h^{q3}, u_z\}$ to be approximated, with $q1 = 1,\ldots,50$; $q2 = 1,\ldots,50$; $q3 = 1,\ldots,50$; $z = 125000$. This grid partition is carried out in the state domain $D = [-196.5,78.5] \times [-50,50] \times [-3,2]$ and the optimal control actions at the 125000 grid points are computed thanks to MOBY-DIC Toolbox [15], Hybrid Toolbox [16] and Multi-Parametric Toolbox [17]. The optimal control action verifies that $u_{min} = -0.3 \le u \le 0.3 = u_{max}$.

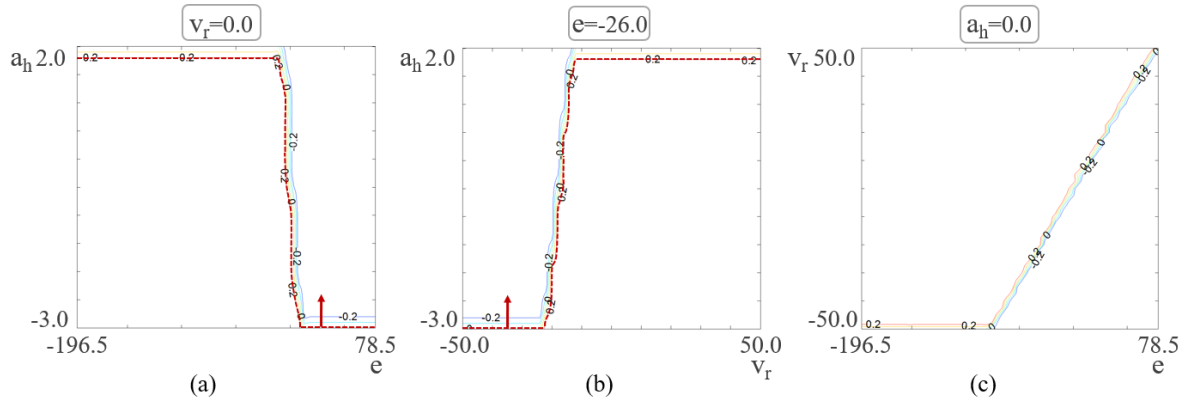

Figure 2. Level curves of the ACC: (a) For the constant value v_r=0.0; (b) For the constant value e=-26.0; (c) For the constant value a_h=0.0

The second design step is to find subsets of points that correspond to level hypersurfaces, $\mathcal{L}_{qu} \subset \mathbb{X}$. Each level hypersurface contains N_{qu} points, denoted as $(e_{lu}, v_{rlu}, a_{hlu}, u_{qu})$ with $l_u = 1, \dots, N_{qu}$, verifying that $|u_{qu} - C_{qu}| \leq \varepsilon_0$ (with ε_0 small). Five level hypersurfaces $\{\mathcal{L}_1, \dots, \mathcal{L}_5\}$ were analyzed, from $C_1 = -0.2$ to $C_5 = 0.2$ in steps of 0.1, since $u_{min} = -0.3 \leq u \leq 0.3 = u_{max}$. Fig. 2 shows examples of these level hypersurfaces in the optimal explicit PWA controller obtained for the ACC system.

A. Selection of the input variable to be separated

The objective of the next design steps is to select one of the n state variables, x_i, to be separated from the rest (in the ACC example, there are three state variables), so as to find a system with the structure of Fig. 3(a), able to approximate the set of points $\mathbb{X} = \{e^{q1}, v_r^{q2}, a_h^{q3}, u_z\}$. According to Equation (4), the aim is to express $u(x)$ as:

$$u(x) = f_{1j} \cdot \left(x_i^{lev} - x_i\right) + g_{1j} =$$
$$f_{1j} \cdot x_i^{lev} - f_{1j} \cdot x_i + g_{1j}, \text{ with } \left(x_i^{lev} - x_i\right) \in I_1^{(j)} \quad (10)$$
$$\text{and } x_i^{lev} = \mathcal{F}_1(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)$$

If Equation (10) is verified, it is possible to express the points $\{x_{1lu}, \dots, x_{nlu}, C_{qu}\}$ of the level hypersurface, $\mathcal{L}_{qu}$, as:

$$x_{ilu} = x_{ilu}^{lev} + \frac{g_{1j} - C_{qu}}{f_{1j}},$$
$$\text{with } x_{ilu}^{lev} = \mathcal{F}_1\left(x_{1lu}, \dots x_{(i-1)lu}, x_{(i+1)lu}, \dots x_{nlu}\right) \quad (11)$$
$$\text{and } \frac{g_{1j} - C_{qu}}{f_{1j}} \text{ a constant for the interval } I_1^{(j)}$$

The first condition to check if Equations (10) or (11) are verified is that $\mathcal{F}_1(.)$ should be a well-defined function, which requires it cannot associate different outputs to the same input. Hence, there should not be equal points $\left(x_{1lu}, \dots x_{(i-1)lu}, x_{(i+1)lu}, \dots x_{nlu}\right)$ in $\mathcal{L}_{qu}$ as a necessary condition to select x_i, because $\mathcal{F}_1(.)$ would be multivalued otherwise. Formally:

$$\forall l_{uv}, l_{uw} \in \{1, \dots, N_{qu}\},$$
$$\left(x_{1luv}, \dots x_{(i-1)luv}, x_{(i+1)luv}, \dots x_{nluv}\right) \neq \quad (12)$$
$$\left(x_{1luw}, \dots x_{(i-1)luw}, x_{(i+1)luw}, \dots x_{nluw}\right)$$

The variable e does not meet this condition as can be seen in Fig. 2(a). For example, in the level hypersurface $\mathcal{L}_5$ ($|u - 0.2| \leq \varepsilon_0$) there are several points with values $v_r = 0.0$ and $a_h = 1.6$. The same happens with v_r. As shown in Fig. 2(b), $\mathcal{L}_5$ has several points with values $e = -26.0$ and $a_h = 1.6$. Hence, the variables e and v_r cannot be selected.

The second condition to check is that all the level hypersurfaces can be represented by the same function $\mathcal{F}_1(.)$ displaced in parallel to the axis x_i. This happens in Fig. 2(a) and 2(b), because the level highlighted can be seen displaced in parallel to the axis a_h. To check this quantitatively, the level hypersurface, $\mathcal{L}_{min}$, with the minimum number of points, N_{min}, is selected to evaluate if the coordinates corresponding to the state variables x_j ($j = 1$ to n; $j \neq i$) in the points of $\mathcal{L}_{min}$ also appear in the points of any other level hypersurface, $\mathcal{L}_u$. Formally (with ε_1 small):

$$\forall lr \in \{1, \dots, N_{min}\} \exists ls * \in \{1, \dots, N_u\} / |x_{jlr} - x_{jls*}| \leq \varepsilon_1, \text{ with } j = 1, \dots, i-1, i+1, \dots, n \quad (13)$$

It can be seen that the curves in Fig. 2(a) have the same grid points of e and that the curves in Fig. 2(b) have the same grid points of v_r.

The third condition is that the differences of the N_{min}

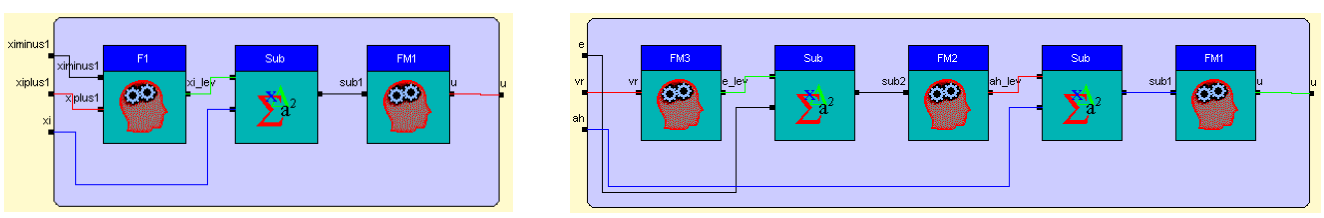

Figure 3. Hierarchical structure for the ACC with Xfuzzy tool *xfedit*: (a) First decomposition; (b) Final PWAH structure.

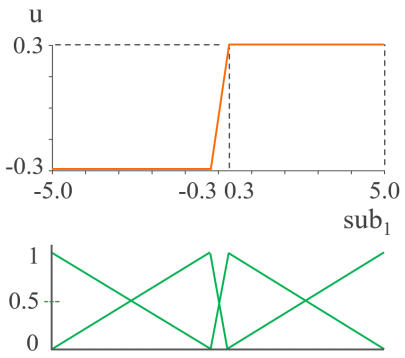

Figure 4. Behavior of the module FM_1 of Fig. 3 and their triangular membership functions, after its design with Xfuzzy tools.

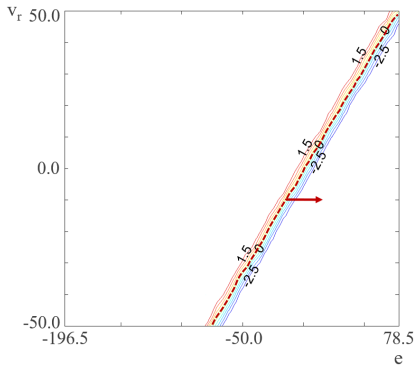

Figure 5. Level curves of the block $\mathcal{F}_1$ in the ACC.

values of the coordinates corresponding to the state variables x_i in each pair of level hypersurfaces, $\mathcal{L}_{min}$ and $\mathcal{L}_u$, should be equal or very similar, that is (with ε_2 small):

$$\exists c \in \mathbb{R} / |(x_{ilr} - x_{ils*})| = c + \varepsilon_2$$
$$\forall lr \in \{1, \dots, N_{min}\} \text{ and } ls * \in \{1, \dots, N_u\} \quad (14)$$

In the examples of Fig. 2(a) and 2(b), it can be seen that the difference between two curves along the axis a_h is approximately the same. Hence, all the conditions from (12) to (14) are met by the variable a_h.

If the above conditions are fulfilled by several state variables, x_i and others x_j, the level hypersurface $\mathcal{L}_{eq}$ corresponding to the control action in the equilibrium state, $u = 0$, is analyzed in detail (it will also be used to define the module $\mathcal{F}_1$). If the data of this hypersurface are $(x_{1leq}, \dots, x_{nleq}, u_{leq})$ with $l_{eq} = 1, \dots, N_{eq}$, the variable selected to be separated, x_i, is that whose percentage of universe of discourse is the least covered by that hypersurface. Formally:

$$\frac{\left|\max_{leq}\{x_{ileq}\} - \min_{leq}\{x_{ileq}\}\right|}{|x_{imax} - x_{imin}|} \leq \frac{\left|\max_{leq}\{x_{jleq}\} - \min_{leq}\{x_{jleq}\}\right|}{|x_{jmax} - x_{jmin}|} \quad \forall j \neq i \quad (15)$$

B. Design of the SISO fuzzy modules

Once the state variable x_i (the variable a_h) has been selected to be separated, decomposing the system in the structure of Fig. 3(a), the next objective is to design the module FM_1. For that purpose, the level hypersurface $\mathcal{L}_{eq}$ corresponding to the control action in the equilibrium state, $u = 0$, is used to define the module $\mathcal{F}_1$. It is assumed that $\mathcal{F}_1(0) = 0$, because the state variables and control action should be zero in the equilibrium. This is equivalent to assume that if $sub_1 = x_{ileq}^{lev} - x_{ileq} = 0 \in I_1^{(eq)}$, then $u = f_{FM1}(sub_1) = f_{FM1}(0) = 0$. Therefore, the points of the level hypersurface $\mathcal{L}_{eq}$ verify that:

$$x_{ileq} = x_{ileq}^{lev} = \mathcal{F}_1(x_{1leq}, \dots x_{(i-1)leq}, x_{(i+1)leq}, \dots, x_{nleq}) \quad (16)$$

with $leq = 1, \dots, N_{eq}$

Since the level hypersurfaces can be represented by the same function, $\mathcal{F}_1$, displaced in parallel to x_i, the points in the level hypersurfaces $\mathcal{L}_{qu}$ verify that (according to Equations (11), (14) and (16)):

$$x_{ils*} = x_{ils*}^{lev} + \frac{g_{1ju} - C_{qu}}{f_{1ju}} = x_{ileq*} + b_{1ju} \quad (17)$$

Therefore, if $sub_1 = (x_{ileq*} - x_{ils*}) \in I_1^{(j)}$ then $u = f_{FM1}(-b_{1ju}) = C_{qu}$. Hence, the module FM_1 in Fig. 3(a) is adjusted to minimize the sum of squared errors obtained when considering the points $sub_1 = (x_{ileq*} - x_{ils*})$ as inputs and C_{qu} as desired output.

For that objective, the univariate PWA function provided by the module FM_1 is obtained by the linear combination of triangular membership functions, $\varphi(sub_1)$, that cover the input universe of discourse as a partition of unity (that is, the sum of all the membership functions evaluated at every input value is always the unity):

$$f_{FM1}(sub_1) = \sum_{i=1}^{P} w_n \cdot \varphi(sub_1) \quad (18)$$

The algorithm employed to adjust the number, P, and location of the triangular membership functions is based on the Incremental Grid algorithm, as explained in [18]. The weights, w_n, of the membership functions are adjusted to minimize the sum of squared errors by using second-order algorithms, as explained in [19].

In the case of the ACC system, the partition of unity obtained is shown at the lower part of Fig. 4, and the PWA function obtained for the module FM_1 is shown at the upper part of Fig. 4.

C. Subsequent decompositions

Once the variable a_h is separated and the module FM_1 is designed, the methodology repeats the steps in order to explore if the block $\mathcal{F}_1$ can also be decomposed. The objective is to find if another block, $\mathcal{F}_2$, and another SISO module, FM_2, can be introduced. In the case of the ACC system, the block $\mathcal{F}_2$ is finally approximated by another module, FM_3, so as to obtain the structure in Fig. 3(b).

The new points to approximate by the decomposition of the block $\mathcal{F}_1$ are the subset of $\mathbb{X}$ corresponding to the level

hypersurface $\mathcal{L}_{eq}$ where the control action is $u = 0$. This subset is represented as $\mathbb{X}_1 = \{e_{leq}^{q11}, v_{rleq}^{q21}, a_{hleq}\}$.

Since $a_{h,min} = -3 \leq a_h \leq 2 = a_{h,max}$, nine level surfaces were analyzed. They can be seen in Fig. 5.

The state variables e and v_r both meet the conditions expressed by Equations (12) to (14) in the nine level surfaces. By applying Equation (15), it is verified that the percentage of universe of discourse covered by e is 55%, while the percentage covered by v_r is 99%. Therefore, the variable selected to be separated is e, as illustrated in Fig. 3(b). The level curves in Fig. 5 can be seen as a function $\mathcal{F}_2$ (highlighted) displaced in parallel to the axis e.

To design the module FM_2, the level surface corresponding to $a_h = 0$ is used, following the same steps commented above for the design of the module FM_1. Similarly, the module FM_3 is designed using the level curve corresponding to $e = 0$.

The final step is to apply again the second-order Levenberg-Marquardt algorithm to readjust the weights of the membership functions of all the SISO modules found by the methodology, so that the final implementation reduces further the sum of squared errors. The hierarchical structure designed, provides a good trade-off between high simplicity and small approximation error (the RMSE obtained is 0.31%).

IV. The use of Xfuzzy CAD tools

Xfuzzy environment [19] allows designing complex fuzzy systems, from its initial description to its final implementation [20].

Xfuzzy description tool (*xfedit*) was used to describe the several systems analyzed during the design. Fig. 3 shows part of the *xfedit* graphical user interface (GUI) employed to describe the first (Fig. 3(a)) and the final (Fig. 3(b)) systems analyzed. The tool allows connecting several modules in cascade and the modules can be defined as operator (crisp) blocks (in this case subtraction blocks) and as rule bases (SISO or MISO modules).

Xfuzzy data mining tool (*xfdm*) allows extracting rule bases from data files. Among several grid- and clustering-based algorithms, *xfdm* includes Incremental Grid algorithm as one of the grid-based algorithms [18]. In particular, *xfdm* allows extracting SISO fuzzy modules based on the linear combination of triangular membership functions covering the input universe of discourse as a partition of unity. The

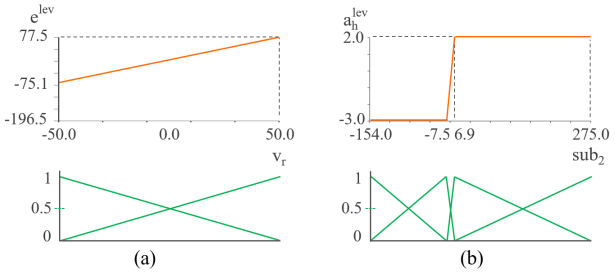

Figure 6. Behavior of the modules (a) FM_3 and (b) FM_2 of Fig. 3 and their triangular membership functions, after their design with Xfuzzy

modules FM_3, FM_2 and FM_1 were designed with *xfdm* from their corresponding data files.

Xfuzzy supervised learning tool (*xfsl*) has multiple algorithms based on gradient, second-order methods and algorithms without derivatives to minimize objective functions (in particular the sum of square errors) [19]. One of them is Levenberg-Marquardt algorithm, which was applied in this methodology to adjust the parameters of all the SISO fuzzy modules. The final behavior of the three modules can be seen in Fig. 4 and Fig. 6, as well as their associated triangular membership functions.

To verify that the system behavior is correct, Xfuzzy environment offers three tools. The tool *xfplot* plots graphically the output versus the inputs. It allows both two- and three-dimensional representations. The monitoring tool, *xfmt*, shows the values of the different internal variables for a given set of inputs. The tool *xfsim* performs closed-loops simulations of the controller with the plant (real or modeled), allowing to study the performance of the controller.

Besides the graphical user interfaces, all Xfuzzy CAD tools employ a common specification language named XFL3, which is user-friendly. In order to translate the XFL3-based descriptions to C, C++, or Java code so as to embed the designed controller into a software implementation, Xfuzzy offers the tools *xfc*, *xfcpp*, and *xfj*. For hardware implementation, Xfuzzy offers two tools: *xfsg* and *xfvhdl*. The first of them, *xfsg*, based on the use of Xilinx's System Generator tool on Matlab, allows to generate, from the controller description, a Simulink model capable of implementing the system in a Xilinx FPGA (Field Programmable Gate Array). The second one, *xfvhdl*, facilitates the hardware synthesis through the generation of synthesizable VHDL code over FPGAs or ASICs (Application Specific Integrated Circuits).

The hierarchical fuzzy controller designed for the ACC problem was implemented in a FPGA. In this case, another advantage of PWAH systems is that the first and the last SISO fuzzy modules can be further simplified if they can be merged with the input and output signal conditioning circuitry, as happens in this example. As can be seen in Fig. 6(a), the module FM_3 applies a linear transformation to the input v_r, so that it can be performed by the input signal conditioning, which has to transform the input values to the range [0,1], in order to implement the hierarchical controllers with fixed-point arithmetic in the FPGA. In addition, as shown in Fig. 4, the module FM_1 applies also a linear transformation to its inputs, which is saturated for values that go beyond the interval [0,1]. This is achieved simply by using the output signal as an unsigned type signal with N bits and the decimal dot in the N position, applying saturation outside that range and performing the lineal transformation with the output signal conditioning circuitry. Therefore, the resulting hierarchical controller only requires two adders and one multiplier. Using a Xilinx Spartan-3AN (XC3S700AN) FPGA, with 16 bits of precision for inputs and outputs, the designed controller consumes 47 slices (approximately 0.8% of the available slices on the FPGA) and uses only 1 of the 20 multipliers in the FPGA. The controller has a latency of one clock cycle and is able to operate at a maximum frequency of 753MHz.

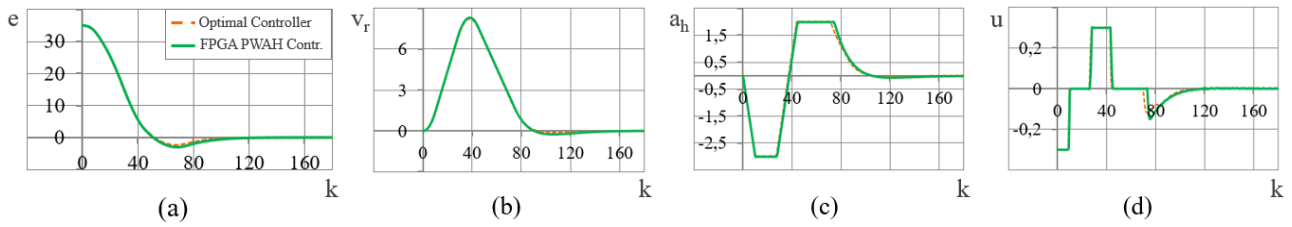

Figure 7. Comparison of the optimal controller (simulation) with the PWAH controller implemented in a FPGA (hardware-in-the-loop) for the ACC

Table II compares the implementation results of several PWA controllers reported in the literature for the ACC problem. All the controllers (PWAG and PWAS in [14], and PWAL in [4]) are implemented in a Xilinx FPGA Spartan 3AN (XC3S700AN), like the PWAH controller described herein. The latency values are provided for a frequency of 20MHz. It can be seen that the PWAH controller designed with the proposed methodology is the fastest, occupies the least percentage of slices, and does not require any memory.

The hardware implementation of the controller in the FPGA was co-simulated with a software description of the ACC plant in Matlab-Simulink. Several simulation results are shown in Fig. 7. The control achieved by the PWAH system is so similar to the optimal control that it is difficult to differentiate one from the other.

V. Conclusions and Future Work

It is not easy to find an adequate hierarchical decomposition of an explicit MPC control law that is given as optimal reference, without the application of a methodology and the aid of CAD tools. This paper has shown how the CAD tools of Xfuzzy environment facilitate the design steps of the methodology to approximate explicit MPC control laws as hierarchical fuzzy controllers, in particular piecewise-affine hierarchical (PWAH) controllers, which consist of a cascade of SISO fuzzy modules with a PWA behavior. The versatility and power of Xfuzzy CAD tools allow an easy and fast design, which starts with numerical data files and finishes with the software or hardware description of the designed controller. This has been illustrated with the design and FPGA implementation of a hierarchical controller for the ACC problem. It performs very close to the optimal explicit MPC controller, using the least resources compared to other reported implementations. The use of SISO modules with another piecewise polynomial behavior will be the object of future work.

TABLE II. Comparison of PWA Controllers for the ACC

Controller	Slices (%)	Latency (µs)	Clock cycles	Mem. (KB)	Mult.
PWAG	87	5.4	108	3.3	1
PWAS(serial)	31	0.4	8	11.5	1
PWAS(parallel)	95	0.15	3	57.6	5
PWAL	5	1.65	33	1.06	8
PWAH	0.8	0.05	1	0	1

References

[1] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems", *Automatica*, vol. 38, no. 1, pp. 3-20, 2002.

[2] A. Bemporad, "A multiparametric quadratic programming algorithm with polyhedral computations based on nonegative least squares", *IEEE Trans. Autom. Control*, vol. 60, no. 11, pp. 2892-2903, 2015.

[3] A. Oliveri, A. Oliveri, T. Poggi, and M. Storace, "Circuit implementation of piecewise-affine functions base on a binary search tree", in *Proc. Eur. Conf. Circuit Theory and Design (ECCTD'09),* 2009, pp. 145-148.

[4] M.C. Martínez-Rodríguez, P. Brox, and I. Baturone, "Digital VLSI implementation of piecewise-affine controllers based on lattice approach"', *IEEE Trans. Control Syst. Technol.,* vol. 23, no. 3, pp. 842-854, 2015.

[5] M. Storace, and T. Poggi, "Digital architectures realizing piecewise-linear multivariate functions: Two FPGA implementations"', *Int. J. Circuit Theory and Appl.*, vol. 39, no. 1, pp. 1-15, 2011.

[6] F. Comaschi, B.A.G. Genuit, A. Oliveri, W.P.M.H. Heemels, and M. Storace, "FPGA implementations of piecewise affine functions based on multi-resolution hyperrectangular partitions"', *IEEE Trans. Circuits Syst. I: Reg. Papers*, vol. 59, no. 12, pp. 2920-2933, 2012.

[7] A. Gersnoviez, M. Brox, and I. Baturone, "High-Speed and Low-Cost Implementation of Explicit Model Predictive Controllers", *IEEE Trans. Control Syst. Technol.,* vol. 27, no. 2, pp. 647-662, 2019.

[8] G.V.S Raju, J. Zhou, and R.A. Kisner, "Hierarchical fuzzy control", *Int.l J. Control*, vol. 54, pp. 1201-1216, 1991.

[9] I. Baturone, S. Sánchez-Solano, A. Gersnoviez, and M. Brox, "An automated design flow from linguistic models to piecewise polynomial digital circuits", in *Proc. 2010 IEEE Int. Symp. on Circuits and Systems (ISCAS'10)*, pp. 3317-3320, 2010.

[10] S. Aja-Fernández, and C. Alberola-López, "Matrix modeling of hierarchical fuzzy systems", *IEEE Trans. Fuzzy Syst*., vol. 16, no. 3, pp. 585-599, 2008.

[11] M.G. Joo, and T. Sudkamp, "A method of converting a fuzzy system to a two-layered hierarchical fuzzy system and its run-time efficiency", *IEEE Trans. Fuzzy Syst*., vol. 17, no. 1, pp. 93-103, 2009.

[12] B. Mutlu, E.A. Sezer, and M.A. Akcayol, "End-to-end hierarchical fuzzy inference solution", *in Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE'18)*, 2018.

[13] G.J.L. Naus, J. Ploeg, M.J.G. Van de Molengraft, W.P.M.H. Heemels, and M. Steinbuch, "Design and implementation of parameterized adaptive cruise control: An explicit model predictive control approach"', *Control Eng. Pract,* vol. 18, no. 8, pp. 882-892, 2010.

[14] A. Oliveri, G.J.L. Naus, M. Storace, and W.P.M.H. Heemels, "Low-complexity approximations of PWA functions: A case study on adaptive cruise control"', *in Proc. Eur. Conf. Circuit Theory and Design (ECCTD'11)*, pp. 669-672, 2011.

[15] MOBY-DIC Toolbox, Available at: http://ncas.dibe.unige.it/software/MOBY-DIC_Toolbox

[16] Hybrid Toolbox, Available at: http://cse.lab.imtlucca.it/~bemporad/hybrid/toolbox

[17] Multi-Parametric Toolbox, Available at: http://people.ee.ethz.ch/~mpt/3

[18] I. Baturone, F.J. Moreno-Velo, A. Gersnoviez, "Identifying fuzzy systems from numerical data with Xfuzzy", *in Proc. Conf. Eur.Soc. Fuzzy Logic and Technol. (EUSFLAT'05)*, pp. 1257-1262, 2005.

[19] Xfuzzy. Fuzzy logic design tools. Available at: http://www.imse-cnm.csic.es/Xfuzzy

[20] F.J. Moreno-Velo, I. Baturone, R. Senhadji, and S. Sánchez-Solano, "Tuning complex fuzzy systems by supervised learning algorithms", *in Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE'03)*, pp. 226-231, 2003.