Robust Evolving Cloud-based PID Control Adjusted by Gradient Learning Method

Igor Škrjanc, Sašo Blažič
Faculty of Electrical Engineering
University of Ljubljana
igor.skrjanc@fe.uni-lj.si, saso.blazic@fe.uni-lj.si

Plamen Angelov
School of Computing and Communications
Lancaster University, United Kingdom
p.angelov@lancaster.ac.uk

Abstract—In this paper an improved robust evolving cloud-based controller (RECCo) for a class of nonlinear processes is introduced. The controller is based on parameter-free premise (IF) part. The consequence in this case is given in the form of PID-type controller. The three adjustable parameters of PID controller are updated on-line with a stable adaptation mechanism based on Lyapunov approach such that the output of the process tracks the desired model-reference trajectory. The proposed algorithm has also ability to add new rules or new clouds when this is necessary to improve the whole behaviour of the controlled process. This means that RECCo controller evolves the control structure and adjusts at the same time the parameters of the controller in an on-line manner, while performing the control of the plant. This approach is an example of almost parameter-free approach, because it does not use any off-line pre-training nor the explicit model of the plant and requires almost no parameter tuning. The proposed algorithm is tested on an artificial nonlinear first-order process and on a simulated hydraulic plant.

I. INTRODUCTION

The PID controllers are widespread because their simple structure, easy implementation and robustness in operation. They are mainly used for control of (almost) linear processes or operate in a small neighbourhood of an operating point where linear approximation is adequate. Different schemes of incorporating nonlinear techniques in the PID controller design emerged to improve the control in the case of nonlinear processes [1], [2], [3], [4]. A lot of modern techniques have been presented for auto-tuning of controllers in batch mode [5] mostly using genetic algorithms [6], [7], or neural networks [8], [9] off-line. However, there is no guarantee that pre-trained parameters have a satisfactory performance in online applications specially when the environment or the controlled object are influenced by changes. This problem has been elaborated in several approaches where different methods for online adaptation for the class of nonlinear processes have been proposed [10], [11], [12], [13], [14], [15], [16],[17], [18], [19], [20].

The evolving fuzzy rule-based controllers [21], [22] enable the construction of the controller structure (fuzzy rules, fuzzy sets, membership functions, etc.) online from the data stream. The main advantage of the proposed method is the fact that no off-line tuning is needed, it has no parameters in the antecedent part of the rules and the control scheme is suitable for a class of nonlinear processes. Moreover, the proposed control scheme can start with just a few parameters which are needed to design the clouds and to adapt the controller gains, with no information about the clouds and with no information about the controller gains. The basic topology of the control scheme, like the input domain of the clouds, and the structure in the consequent part should be predefined. The topology and the structure of the controller is then modified online based on the data stream during the closed loop operation of the controlled system.

The proposed control scheme introduces two main mechanisms: the adaptation of control parameters in the consequent part of the fuzzy rules and the evolution and modification of the controller structure. The advantage of using clouds instead of more frequently used membership functions is that it does not require any membership functions to be defined for the antecedent part nor any fuzzy operators. It still has a linguistic form and enable modelling of non-linear behaviour. The proposed cloud-based approach resembles fuzzy one in terms of the defuzzification. The three main structures of fuzzy systems: Mamdani, Takagi-Sugeno (TS) and Cloud-based can be represented as a set of fuzzy rules of the form: if (antecedent) then (consequent) with the Ts fuzzy rule-type. For more detail, please refer to [23], [24]. Some interesting control implementations of the proposed cloud-based approach have been reported in [9], [5], [25], [23], [22], [26].

In this paper, the main contribution lies in a simplified mechanism for evolution of the controller topology. According to cloud-based fuzzy system paradigm the clouds are added based on the local and global density which is used to define the relative membership to a particular cloud. The density takes into account the distances to all previous data samples and can be calculated recursively. In our new approach the evolving part is based only on the local density and the global
one is not needed any longer. The PID controller gains are adapted using gradient-based learning although other types of optimisation are also possible [27], [19]. This simplifies the controller structure and makes it much more robust. The adaptation and the evolution mechanism can both be switched off after a successful learning phase and the approach can be therefore seen as self-tuning. It would be also of great practical use to have the functionality to switch them on only when the performance of the controlled close-loop systems fails to meet the required criteria.

The remainder of this paper is organized as follows. In section II the new simplified fuzzy rule based system is introduced. The consequence part of the rule-base structure is presented in section II-A. The methodology used for the adaptation of the controller gains in the consequence and the robust modification of the adaptive laws are given in section III. In section IV we illustrate the proposed structure and evolution mechanism. In section V, the simulation examples are presented as a proof of concept for the proposed methodology. Finally, the conclusions are drawn in section VI.

II. Structure of the cloud-based controller

The main feature of the proposed cloud-based design framework [24] is the use of non-parametric antecedents. It does not require, unlike the other fuzzy approaches, any explicit definition of fuzzy sets for each input variable. This framework applies the concepts of fuzzy data clouds and relative data density to define antecedents. The density of the data in the input data space can be easily computed recursively from the streaming data online.

The data clouds are defined as subsets of data samples which are close in the data space. The density includes the information of all previous data samples, directly and exactly. Each data sample belongs to each cloud with a different degree of density, described as \( \gamma \in [0, 1] \). This implies the fuzziness into the cloud based framework. The clouds are different from traditional clusters and have no specific shapes and thereby do not require the definition of boundaries.

For each cloud one rule is formed. The whole set of clouds forms a rule base with \( N \) rules of the following form

\[
R^i : \text{IF } (\vec{x} \sim X^i) \text{ THEN } (u^i) \tag{1}
\]

where \( \sim \) denotes the fuzzy membership expressed linguistically as “is associated with”, \( X^i \in \mathcal{R}^n \) is the \( i \)-th data cloud defined in the input space, \( \vec{x} = [x_1, x_2, \ldots, x_n]^T \) is the controller’s input vector and \( u^i \) is the control action defined by the \( i \)-th rule.

Using the rule-based structure given in (1) can describe complex, generally non-linear, non-stationary, non-deterministic systems. The systems that can be only observed through their inputs and outputs.

The degree of association between the data sample \( \vec{x}_k \) and corresponding cloud \( X^i \) is measured by the normalized relative density as follows:

\[
\lambda^i_k = \frac{\gamma^i_k}{\sum_{j=1}^{N} \gamma^j_k} \quad i = 1, \ldots, N \tag{2}
\]

where \( \gamma^i_k \) is the local density of the \( i \)-th cloud for that data sample.

The local density is computed by a chosen kernel which is defined as the distance between \( \vec{x}_k \) and all the other samples in the cloud, i.e.,

\[
\gamma^i_k = \mathcal{K} \left( \sum_{j=1}^{M^i} d^j_{kj} \right) \quad i = 1, \ldots, N \tag{3}
\]

where \( d^j_{kj} \) denotes the distance between the data samples \( \vec{x}_k \) and \( \vec{x}_j \), and \( M^i \) is the number of input data samples associated with the cloud \( X^i \). The distance is computed as the Euclidean distance, i.e., \( d^j_{kj} = ||\vec{x}_k - \vec{x}_j||^2 \). Nonetheless, any other type of distance could also be used [24].

The density is defined by Cauchy kernel as follows

\[
\gamma_k = \frac{1}{1 + \frac{\sum_{j=1}^{k-1} (d_{kj})^2}{k}} \tag{4}
\]

which is in recursive form written as follows [28]:

\[
\gamma^i_k = \frac{1}{1 + ||x_k - \mu_k||^2 + \sigma_k - ||\mu_k||^2} \tag{5}
\]

where \( \gamma^i_k \) denotes the local density of \( k \)-th data sample to the \( i \)-th data cloud, \( \Sigma_k \) denotes the mean-square length of the data vectors in the cloud and is the following

\[
\Sigma_k = \frac{k-1}{k} \Sigma_{k-1} = \frac{1}{k} ||x_k||^2 \tag{6}
\]

with initial condition \( \Sigma_1 = ||x_1||^2 \) and the mean value, \( \mu \) is straightforward

\[
\mu_k = \frac{k-1}{k} \mu_{k-1} + \frac{1}{k} x_k^2 \tag{7}
\]

with starting condition \( \mu_1 = x_1^2 \).

The density of the current data sample is computed for all clouds. It is than considered that a sample belongs to the cloud or is fully associated with the cloud with the highest local density.

When the consequence of the rule base is calculated, the defuzzification computed as the weighted average is used

\[
u_k = \sum_{i=1}^{N} \lambda^i_k u^i = \frac{\sum_{i=1}^{N} \gamma^i_k u^i}{\sum_{i=1}^{N} \gamma^i_k} \tag{8}
\]

where \( u^i \) denotes the \( i \)-th rule consequent.
A. The consequence part of the rule-base structure

The main goal of the controller is to bring the process output from its current value to the desired reference value as soon as possible. Very often the reference model is introduced into the closed-loop system that represents the desired dynamics of the controlled systems [29]. The simplest choice is to use a linear reference model of the first order. The model-reference output $\hat{y}$ is defined using the following equations

$$\hat{y}_{k+1} = \alpha_r \hat{y}_k + (1 - \alpha_r) r_k \quad 0 < \alpha_r < 1$$

where $\alpha_r$ is the pole of the first order filter and define the first order dynamics.

The process output $y_k$ is compared to the reference model $\hat{y}_k$ to obtain the tracking error $\varepsilon$ which is given as follows:

$$\varepsilon_k = y_k - \hat{y}_k$$

As already mentioned, the proposed approach is widely used in different control laws with different rule consequents. In the proposed approach, the PID-based rule consequents is proposed. The PID-type control consequent has the following form:

$$u_i^k = P_i^k \varepsilon_k + I_i^k \sum_{n=0}^{k-1} \varepsilon_n + D_i^k \Delta \varepsilon_k + R_i^k \quad i = 1, \ldots, N$$

where $P_i^k$, $I_i^k$, $D_i^k$ and $R_i^k$ are controller gains, $R_i^k$ is compensation of the operating point for each cloud $i = 1, \ldots, N$, where $\sum \varepsilon_n$ and $\Delta \varepsilon_k$ denote respectively the discrete-time integral and derivative of the tracking error.

The controller parameters $P_i^k$, $I_i^k$, $D_i^k$ and $R_i^k$ will be tuned by means of rule consequents adaptation. The approach offers the possibility of implementing several subsets of PID-based controllers such as P, PI, PD etc.

III. ADAPTATION OF THE CONTROLLER GAINS

When discussing the adaptive laws, the assumption of known process gain sign $G_{\text{sign}} = \pm 1$ has to be made. Therefore, the combination of the error at the process output and the sign of the monotonicity of the process w.r.t. the control signal, provides information about the right direction in which to adapt the controller gains in the rule consequents to achieve the local control objective [11].

The controller gains are in the normal circumstances adapted as follows:

$$\Delta P_i^k = \gamma_P G_{\text{sign}} \lambda_i \frac{\varepsilon_k}{1 + r_k^2}$$

$$\Delta I_i^k = \gamma_I G_{\text{sign}} \lambda_i \frac{\varepsilon_k}{1 + r_k^2}$$

$$\Delta D_i^k = \gamma_D G_{\text{sign}} \lambda_i \frac{\varepsilon_k}{1 + r_k^2}$$

where $\gamma_P$, $\gamma_I$, $\gamma_D$, $\gamma_R$ are the adaptive gain for the controller gains, $\lambda_i^k$ is the normalized relative density and $\varepsilon_k$ is defined for $r_k = r_k - y_k$.

The adaptive control law is obtained from Eq. (16) by using gradient descent and having the square of the tracking error as a cost function. The controller gains define the controller gain vector $\theta_i^k = [P_i^k, I_i^k, D_i^k, R_i^k]^T$ and than the adapted parameter by using (16) are defined as:

$$\theta_i^k = \theta_i^{k-1} + \Delta \theta_i^k \quad i = 1, \ldots, N$$

It should be mentioned that parameters keep changing until the tracking error is driven towards 0 and also that only parameters corresponding to the active clouds are adapted while the others are kept constant.

Systems with parameter adaptation are subjected to parameter drift that can lead to performance degradation and, eventually, to system instability [30]. There exist many known approaches to make adaptive laws more robust [31], [32], [33]. We will employ parameter projection, parameter leakage, introduce dead zone into adaptive laws and employ the saturation of the adaptive parameters when the actuator is in saturation.

1) Dead zone in the adaptive law: The adaptation of parameters in the closed loop always presents potential danger to the system stability. The adaptation is driven by an error signal that is always composed of the useful component and the harmful one, which is caused by disturbances and parasitic dynamics. Parasitic dynamic is usually bounded and large error is usually mostly composed of the useful component. The persistent adaptation when the error is small may results in a false adaptation. The idea behind the dead zone in the adaptive law is that the adaptation is simply switched off if the absolute value of the error that governs the adaptation is small [34]:

$$\Delta \theta_i^k = \begin{cases} \Delta \theta_i^k, & |\varepsilon_k| \geq d_{\text{dead}} \\ 0, & |\varepsilon_k| < d_{\text{dead}} \end{cases}$$

2) Parameter projection: The natural way to prevent the parameter drift is projection of the parameters onto a compact set [29]. In the case of the positive plant gain all the consequent parameters should be bounded by 0 from below while upper bound may or may not (if not enough prior knowledge is available) be provided. The adaptive law given by (17) is generalized as follows if the controller gains are projected onto the interval $[\theta, \bar{\theta}]$:

$$\theta_i^k = \begin{cases} \theta_i^{k-1} + \Delta \theta_i^k, & \theta \leq \theta_i^{k-1} + \Delta \theta_i^k \leq \bar{\theta} \\ \theta, & \theta_i^{k-1} + \Delta \theta_i^k < \theta \\ \theta_i^{k-1} + \Delta \theta_i^k > \bar{\theta} \end{cases}$$

3) Leakage in the adaptive law: The idea of leakage is that the discrete integration in the adaptive law in 17 presents the potential danger to the adaptive system and the pole due to the integrator should be pushed inside the unit disc [35] resulting in the adaptive law:

$$\theta_i^k = (1 - \sigma_L)\theta_i^{k-1} + \Delta \theta_i^k \quad i = 1, \ldots, N$$

where $\gamma_P$, $\gamma_I$, $\gamma_D$, $\gamma_R$ are the adaptive gain for the controller gains, $\lambda_i^k$ is the normalized relative density and $\varepsilon_k$ is defined for $r_k = r_k - y_k$. 

The adaptive control law is obtained from Eq. (16) by using gradient descent and having the square of the tracking error as a cost function. The controller gains define the controller gain vector $\theta_i^k = [P_i^k, I_i^k, D_i^k, R_i^k]^T$ and than the adapted parameter by using (16) are defined as:

$$\theta_i^k = \theta_i^{k-1} + \Delta \theta_i^k \quad i = 1, \ldots, N$$

It should be mentioned that parameters keep changing until the tracking error is driven towards 0 and also that only parameters corresponding to the active clouds are adapted while the others are kept constant.

Systems with parameter adaptation are subjected to parameter drift that can lead to performance degradation and, eventually, to system instability [30]. There exist many known approaches to make adaptive laws more robust [31], [32], [33]. We will employ parameter projection, parameter leakage, introduce dead zone into adaptive laws and employ the saturation of the adaptive parameters when the actuator is in saturation.

1) Dead zone in the adaptive law: The adaptation of parameters in the closed loop always presents potential danger to the system stability. The adaptation is driven by an error signal that is always composed of the useful component and the harmful one, which is caused by disturbances and parasitic dynamics. Parasitic dynamic is usually bounded and large error is usually mostly composed of the useful component. The persistent adaptation when the error is small may results in a false adaptation. The idea behind the dead zone in the adaptive law is that the adaptation is simply switched off if the absolute value of the error that governs the adaptation is small [34]:

$$\Delta \theta_i^k = \begin{cases} \Delta \theta_i^k, & |\varepsilon_k| \geq d_{\text{dead}} \\ 0, & |\varepsilon_k| < d_{\text{dead}} \end{cases}$$

2) Parameter projection: The natural way to prevent the parameter drift is projection of the parameters onto a compact set [29]. In the case of the positive plant gain all the consequent parameters should be bounded by 0 from below while upper bound may or may not (if not enough prior knowledge is available) be provided. The adaptive law given by (17) is generalized as follows if the controller gains are projected onto the interval $[\theta, \bar{\theta}]$:

$$\theta_i^k = \begin{cases} \theta_i^{k-1} + \Delta \theta_i^k, & \theta \leq \theta_i^{k-1} + \Delta \theta_i^k \leq \bar{\theta} \\ \theta, & \theta_i^{k-1} + \Delta \theta_i^k < \theta \\ \theta_i^{k-1} + \Delta \theta_i^k > \bar{\theta} \end{cases}$$

3) Leakage in the adaptive law: The idea of leakage is that the discrete integration in the adaptive law in 17 presents the potential danger to the adaptive system and the pole due to the integrator should be pushed inside the unit disc [35] resulting in the adaptive law:

$$\theta_i^k = (1 - \sigma_L)\theta_i^{k-1} + \Delta \theta_i^k \quad i = 1, \ldots, N$$

where $\gamma_P$, $\gamma_I$, $\gamma_D$, $\gamma_R$ are the adaptive gain for the controller gains, $\lambda_i^k$ is the normalized relative density and $\varepsilon_k$ is defined for $r_k = r_k - y_k$. 

The adaptive control law is obtained from Eq. (16) by using gradient descent and having the square of the tracking error as a cost function. The controller gains define the controller gain vector $\theta_i^k = [P_i^k, I_i^k, D_i^k, R_i^k]^T$ and than the adapted parameter by using (16) are defined as:

$$\theta_i^k = \theta_i^{k-1} + \Delta \theta_i^k \quad i = 1, \ldots, N$$

It should be mentioned that parameters keep changing until the tracking error is driven towards 0 and also that only parameters corresponding to the active clouds are adapted while the others are kept constant.

Systems with parameter adaptation are subjected to parameter drift that can lead to performance degradation and, eventually, to system instability [30]. There exist many known approaches to make adaptive laws more robust [31], [32], [33]. We will employ parameter projection, parameter leakage, introduce dead zone into adaptive laws and employ the saturation of the adaptive parameters when the actuator is in saturation.
where $\sigma_L$ defines the extent of the leakage. The introduction of the leakage results in adaptive parameter boundedness.

4) **Interuption of adaptation**: In some cases the dynamics and the real actuator constraints are in a conflict, and that the system can not achieve the tracking of the reference model (in a sufficiently small interval). The persistent tracking error $\varepsilon_k$, which can not be reduced due to the control signal constraints, can lead to the drift of the control parameters. In such case the adaptive law should be interrupted in the following manner:

$$\Delta \theta_{ik} = \begin{cases} \Delta \theta_{ik} & 0 \leq u_{\min} \leq u_k \leq u_{\max}, \\ 0 & \text{else} \end{cases} \quad i = 1, \ldots, N \quad (21)$$

**IV. EVOLVING METHODOLOGY FOR THE RECCO**

In this section the evolving methodology and the pseudo-code of the proposed control algorithm will be given. Firstly, the controller is initialized defining a few constants which can be generally used for a wide class of processes. After this, the same steps are repeated for every incoming data: firstly, the consequents of the current rules are updated according to the error at the process output. Then, a new control action is generated by applying the weighted average of consequent parts as proposed. Finally, the controller’s structure is updated. If the appropriate conditions are satisfied, a new cloud is created; otherwise, the new sample is used to update the information about the data density and the consequent parameters of the controller’s current configuration.

**A. Structure evolution: adding new clouds**

Traditionally adaptive systems concern tuning parameters of the controllers for which the structure has been selected by the designer in the designing phase ([36]). On-line the consequent parameters are adapted in a closed control loop. The self-evolving controllers [21], [25] evolve the structure of the controller together with the parameters adaptation during the operation. The evolving part requires the mechanism for adding new antecedents and fuzzy rules.

The mechanism of fuzzy rules addition given in this paper is much simplified in comparison to the approaches given in [21], [25], [22], [26]. The simplified approach to adding clouds is based on local data density only, and no global density is needed. The first criteria which should be fulfilled is that all local densities $\gamma_k$ are smaller then initially defined $\gamma_{\max}$. This means that $\max \gamma_k < \gamma_{\max}$. And a new cloud can be added after at least $n_{\text{add}}$ data samples from the last update of clouds. The cloud is added only when the current number of clouds $N$ is less then maximal number of clouds $N_{\max}$. Another important parameter is also the number of current data in the cloud denoted as $n^i$. If the data sample is too far from all clouds, i.e. $\max \gamma_k < \gamma_{\min}$, this means that the data should be an outlier and can be neglected.

The pseudo-code of the proposed algorithm in Algorithm 1, where $k_{\text{add}}$ stands for the current time instant $k$ when the previous cloud was added.

**Algorithm 1** Pseudo-code of the RECCO PID control algorithm

1. Initialize (the adaptive parameters): $\gamma_{0}, \sigma_{\text{dead}}, \sigma_{L}$.
2. Initialize (the evolving parameters): $N = 0, \gamma_{\max}, \sigma_{\max}, k_{\text{add}}$.
3. Initialize (the process parameters): $a_r, u_{\min}, u_{\max}, P_0^0, I_0^0, D_0^0, R_0^0$.
4. repeat
   5. Compute: $c_k, \varepsilon_k, \Delta \theta_{ik}^2, k_{\text{add}}$, $\sigma_k^2$.
   6. Compute: $\bar{u}_k = [\varepsilon_k, y_k^T, u_k^T]$.\n   7. if $N = 0$,
      Initialize: $\mu_0^i, \sigma_0^i$ and $N$.
   8. elseif $\max_i \gamma_k^i < \gamma_{\max}$ and $k > n_{\text{add}} + k_{\text{add}}$
      Increment: $N$, store: $k_{\text{add}}$.
      Initialize: $\mu_0^i, \sigma_0^i$.
   9. else Compute: $\gamma_k^i, \mu_k^i, \sigma_k^i$.
    10. Associate sample with cloud and increment $n^i$.
11. end
12. Adaptation of PID controller gains
13. Computation and implementation of control law
14. Measurement of process output
15. until end of data stream

**V. SIMULATION STUDY**

The simulation study of the proposed robust evolving cloud based PID controller is presented in this section. Two different examples are studied. The first one is an artificial nonlinear first order process and the second one is a hydraulic pilot plant of two tanks linked by a valve. Different robust modifications of basic adaptive laws are studied. Main attention is given to the implementation of parameter projection, parameter leakage and introduction of dead zone into the adaptive laws. In all examples it is assumed that no a-prior plant dynamics knowledge is give and used in the controller tuning.

The simulations which are shown here are started from zero fuzzy rules and membership functions and the structure evolve during the process. And, what is the most important, the parameters of the proposed RECCO algorithm are the same in all simulations. In the simulations the dead zone modification is used to make the adaptation robust. The dead zone was chosen as $d_{\text{dead}} = 0.05$. The leakage parameter is for all adaptive gains the same and is equal $\sigma_L = 10^{-6}$. The clouds are added when the maximal value of $\gamma_k^i$, $i = 1, \ldots, N$ is lower than the predefined value $\gamma_{\max} = 0.75$. The number of the clouds depends on the choice of $\gamma_{\max}$ value, which should be in the interval $[0, 1]$. The smaller value of $\gamma_{\max}$ results in lower number of clouds and correspondingly lower quality of the control behavior. All adaptive gains $\gamma_P$, $\gamma_I$, $\gamma_D$ and $\gamma_R$ are equal 0.1. The parameter which defines the minimal number of the samples from the last cloud update before the algorithm add a new cloud is defined as $n_{\text{add}}$ and is equal 20 for all simulations. The process input or the actuator’s interval was given as $[0, 10]$ and the reference-model parameter is defined as $a_r = 0.95$ and the noise $N(0, 0.005)$ is added.
Fig. 1. The reference, model reference and the output signal tracking and the control signal for hydraulic plant process in the whole time interval.

Fig. 2. The parameters of PID controller.

Fig. 3. The clouds in the input space $\vec{x} = [\epsilon_k, y_k]^T$.

Fig. 4. The tracking error $\epsilon_k$ for hydraulic plant process in the whole time interval.

Fig. 5. The reference, model reference and the output signal tracking and the control signal for hydraulic plant process in the starting phase.

to the output. The first order reference-model is used for both process models, even though the actual plant is in the second example of the second order.

In the first simulation example an artificial first order process with quadratic static nonlinearity is studied. The mathematical model is given in the discrete time domain as follows:

$$y^p_k = a_p y^p_{k-1} + b_p u_k$$

$$y_k = (y^p_k)^2 + n_k$$

where $y^p_k$ stands for intermediate variable, $a_p = 0.98$, $b_p = 0.01$ are the process parameters and $n_k$ stands for output noise with characteristics $\mathcal{N}(0, 0.005)$. The sampling period in the discrete implementation is 2 seconds.

The plant exhibits a huge nonlinearity in the static gain. The clouds and their input are defined to enable dealing with the nonlinearity. The input variable to the if part of the controller
is given as $\vec{\varepsilon} = [\varepsilon_k, y_k]^T$, with the model-reference trajectory $y_k$ and the tracking error $\varepsilon_k$.

Fig. 1 shows the reference, the model reference and the output signal in the upper plot and the control signal in the lower part with PID control for hydraulic process. The adaptive gains are shown in Fig. 2, where the parameters for all clouds are shown. It is show that the parameters converge. The clouds generated during the evolving procedure are shown in Fig. 3, where 4 clouds were generated. The tracking error $\varepsilon_k$ is shown in Fig. 4. The model reference tracking is suitable and the parameters of the adaptive law do converge and enable a reasonable performance. At the end a close look to the control behavior of the starting phase and the finishing phase (of the simulation interval) is given in Fig. 5 and Fig. 6, respectively.

The second simulation study is realized for a hydraulic pilot plant of two tanks linked by a valve. The first tank ($V_1$) has the inlet mass flow and the outlet flow into the second tank that is on the same height. The second tank ($V_2$) has also an outlet valve. The input of the process is the mass flow into the first tank $\phi_1$ and the output is the height of the fluid in the second tank $h_2$. The mathematical model the plant in continuous time domain is as follows:

$$\rho S_1 \dot{h}_1 = \phi_1 - k_{v1} \sqrt{|h_1 - h_2|} \text{sign}(h_1 - h_2)$$
$$\rho S_2 \dot{h}_2 = k_{v1} \sqrt{|h_1 - h_2|} \text{sign}(h_1 - h_2) - k_{v2} \sqrt{h_2}$$

(24)

where $S_1 = 0.05$ (the units of the plant parameters are omitted) and $S_2 = 0.05$ are cross-sections of the tanks, $h_1$ and $h_2$ are fluid heights in the tanks, $\rho = 1000$ is the fluid density, while $k_{v1} = 3$ and $k_{v2} = 1$ are constants of the valves.

The sampling period in the discrete implementation is 2 seconds.

The plant exhibits a quit big nonlinearity in the static gain and in the time constants of the process. The reference signal which is going through a bride operating regime is chosen to show the ability of learning. The clouds and their input are defined to enable dealing with the nonlinearity. From that reason the input variable to the if part of the controller is given as $\vec{\varepsilon} = [\varepsilon_k, y_k]^T$, with the model-reference trajectory $y_k$ and the tracking error $\varepsilon_k$.

Fig. 7 shows the reference, the model reference and the output signal in the upper plot and the control signal in the lower part with PID control for hydraulic process. The adaptive gains are shown in Fig. 8, where the parameters for all clouds are shown. It is show that the parameters converge. The clouds generated during the evolving procedure are shown in Fig. 9, where 4 clouds were generated. The tracking error $\varepsilon_k$ is shown in Fig. 10. The model reference tracking is suitable and the parameters of the adaptive law do converge and enable a reasonable performance. At the end a close look to the control behavior of the starting phase and the finishing phase (of the simulation interval) is given in Fig. 11 and Fig. 12, respectively.
Fig. 9. The clouds in the input space $\vec{x} = [\epsilon_k, y_k^r]^T$.

Fig. 10. The tracking error $\epsilon_k$ for hydraulic plant process in the whole time interval.

Fig. 11. The reference, model reference and the output signal tracking and the control signal for hydraulic plant process in the starting phase.

Fig. 12. The reference, model reference and the output signal tracking and the control signal for hydraulic plant process in the finishing phase.

VI. CONCLUSIONS

In this paper, an improved robust evolving cloud-based fuzzy controller (RECCo) is proposed. The proposed structure needs no antecedent parameters. It has been shown that the proposed controller can start with no a-priori knowledge and works adequately for two completely different processes, even in the structure, with the same initial parameters needed to start the control algorithm. The controller performs the evolving algorithm simultaneously together with the control of the plant. The proposed control algorithm is useful for a class of stable nonlinear processes. The algorithm divide the input space of the problem into the clouds and enable the use of different control parameters in each cloud. It enables also the adaptation of the control parameters inside the cloud when the process parameters change during the control. The main advantage of the proposed algorithm lies is a real plug-and-play feature.

REFERENCES


