Applied Soft Computing 48 (2016) 29–38

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Review Article

A practical implementation of Robust Evolving Cloud-based Controller with normalized data space for heat-exchanger plant

Goran Andonovski[a,*], Plamen Angelov[b], Sašo Blažič[a], Igor Škrjanc[a]

[a] Faculty of Electrical Engineering, University of Ljubljana, Slovenia
[b] School of Computing and Communications, Lancaster University, United Kingdom

ARTICLE INFO

Article history:
Received 7 February 2016
Received in revised form 6 May 2016
Accepted 25 May 2016
Available online 23 June 2016

Keywords:
Evolving system
Robust adaptive control
Fuzzy cloud-based system

ABSTRACT

The RECCo control algorithm, presented in this article, is based on the fuzzy rule-based (FRB) system named ANYA which has non-parametric antecedent part. It starts with zero fuzzy rules (*clouds*) in the rule base and evolves its structure while performing the control of the plant. For the consequent part of RECCo PID-type controller is used and the parameters are adapted in an online manner. The RECCo does not require any off-line training or any type of model of the controlled process (e.g. differential equations). Moreover, in this article we propose a normalization of the cloud (data) space and an improved adaptation law of the controller. Due to the normalization some of the evolving parameters can be fixed while the new adaptation law improves the performance of the controller in the starting phase of the process control. To assess the performance of the RECCo algorithm, firstly a comparison study with classical PID controller was performed on a model of a plate heat-exchanger (PHE). Tuning the PID parameters was done using three different techniques (Ziegler–Nichols, Cohen–Coon and pole placement). Furthermore, a practical implementation of the RECCo controller for a real PHE plant is presented. The PHE system has nonlinear static characteristic and a time delay. Additionally, the real sensor's and actuator's limitations represent a serious problem from the control point of view. Besides this, the RECCo control algorithm autonomously learns and evolves the structure and adapts its parameters in an online unsupervised manner.

Contents

* Corresponding author.
E-mail addresses: goran.andonovski@fe.uni-lj.si (G. Andonovski), p.angelov@lancaster.ac.uk (P. Angelov), saso.blazic@fe.uni-lj.si (S. Blažič), igor.skrjanc@fe.uni-lj.si (I. Škrjanc).

http://dx.doi.org/10.1016/j.asoc.2016.05.036
1568-4946/

1. Introduction

Nowadays, control of nonlinear and complex processes is still an active research topic. Besides changing circumstances, process dynamics and complexity of the processes the industrial markets require high and satisfactory performance of the controller. A local linear approximation of the process combined with the classical PID controller provides good results but only in the neighborhood of the linearized operating point while this approach is not suitable for the whole operating range of the nonlinear process.

To solve the problem of nonlinearity the authors in [1] presented a self-tuning method for a class of nonlinear PID control systems based on Lyapunov approach. Another scheme in [2] is presented where the just-in-time learning technique is employed to predict the process dynamics and furthermore, the Lyapunov method for adapting the PID parameters is used. There are many other techniques and methods, for example, in [3] an online adaptation of PID controller using neural networks is proposed and in [4] the genetic algorithm for finding the optimal PID parameters is applied. Also the particle swarm optimization for tuning the parameters of PID controller in [5] is used. Another type of PID controllers are Fractional Order PID (FO PID) controllers that perform better than a classical PID-s [6] but require setting of two additional parameters. Similar to classical ones, tuning of this parameters can be solved by solving an optimization problem [7–9].

Fuzzy systems represent control scheme which is developed to deal with the nonlinear processes and due to their powerful adaptability and nonlinear modeling capability they are widely used in many applications [10–17]. The author of fuzzy sets/systems is Prof. Lotfi A. Zadeh who firstly introduced the theory in [18]. After Prof. Zadeh has introduced the theory of fuzzy sets, Mamdani in [19] published the first fuzzy model based control application on dynamic plant (a model of steam engine). Another fuzzy control system is Takagi–Sugeno (TS) fuzzy approach proposed in [20] that has attracted lots of attention after the publication. The wide popularity and usage of the fuzzy control systems is presented in [21] where a lot of fuzzy control schemes are discussed. Similar to TS fuzzy models a new tensor product (TP) models were developed. One of the advantages of the TP models is that the linear matrix inequality (LMI)-based control design can be applied directly to TP models. Recently, several process control solution using TP models were proposed for different applications [22–25].

Mamdani and TS fuzzy systems are made up of IF-THEN fuzzy rules representing the local linear input/output relations of a nonlinear system. The first part (IF) of the conditional is termed the antecedent, and the second part (THEN) is the consequent. Although, both TS and Mamdani are fuzzy systems, they have some differences, especially in the way how the conditional part is defined. Both fuzzy systems – TS and Mamdani – have a fuzzy antecedent part, while they differ in the consequent part which has the form of a functional (often linear) in the case of TS systems and a form of fuzzy logic in the case of Mamdani systems (see Table 1).

Besides the classical fuzzy rule-based (FRB) systems, TS and Mamdani, Angelov and Yager proposed a new simplified type of FRB system named ANYA in [26]. Moreover, they presented a new concept how the antecedent part is defined. As we have already mentioned above in both classical FRB systems the antecedent part is fuzzy and uses predefined and fixed membership functions of triangular, trapezoidal, Gaussian type, etc. ANYA FRB system extracts the information from the **real data** and form the data *clouds* to define the membership function. The clouds are sets of data that have common properties (they are close to each other in the data space). All data have different degree of memberships to the existing clouds determined by the local density of the data sample to **all** data from the particular cloud.

In [26] the authors distinguished between the clouds and the clusters and they pointed out the main differences between them. In general, the clouds do not require *a priori* information about the total number of membership functions or even an assumption about its form (do not have boundaries). Moreover, data clouds represent **all** previous data samples that are associated with the cloud.

Inspired and motivated by the simplicity of the ANYA FRB system several approaches on process control were developed and tested on different simulation models [27,28] and on a real plant [29]. Firstly, in [27] a new fuzzy controller RECCo (Robust Evolving Cloud-based Controller) was introduced. The main advantage of the RECCo controller is that it does not require any information and knowledge about the controlled process (e.g. in a form of differential equations). Furthermore, it is **initialized** from the first data sample and learns autonomously while performing the control of the plant. Also the structure of the RECCo is not predefined but **evolves** in an online manner during the process control (adding new clouds – fuzzy rules). In [27,29] a new cloud is added according to the global density of the data while in [28,30] a simpler way using local density threshold is proposed. Finally, controller's parameters in the consequent part are also tuned and **adapted** autonomously using stable gradient-based learning method.

In this paper we propose an improvement of the RECCo controller presented in [28]. Our idea is by using the basic knowledge from the controlled process (input and output range, time constant and sampling time) to set/fix the initial parameters required by the algorithm. A new normalized data space is proposed and due to this the evolving parameter γ_{max} can be fixed (γ_{max} defines 'when' a new cloud is added and will be introduced later in more detail). Also the adaptation gain vector $\boldsymbol{\alpha}$ could be calculated using the range of the control variable and the default value. Thus the controller tuning is simplified which makes the approach more appealing for the use in practical applications. Different initial real life scenarios were analyzed and new improved adaptation law with absolute values in the starting phase is proposed to improve the performance of the controller [31]. This improvement speed up convergence and reduce large transients when the initial are far away from the unknown parameters.

In order to show the effectiveness of the proposed controller, we provide several experiments on a real plate heat-exchanger (PHE) plant and on a PHE model. Firstly, we compared the performance of the proposed algorithm RECCo with the classical PID controller on PHE model. The parameters of the PID were tuned using Ziegler–Nichols [32], Cohen–Coon [33] and by pole placement method [34]. Nowadays, the PHE is widely used in many different industries and it is suitable to apply for heating, cooling systems, heat-ventilation-air-condition (HVAC) system, in chemistry, pharmacy, food and beverages industry etc. The basic concept of the PHE is transferring the heat between two liquids (separate circuits) flowing on either side of thin metal plates. The dynamical characteristic of the PHE contains strong nonlinear behavior in gain and time constant and has time delay.

Table 1
A comparison of different types of FRB [26].

	Antecedent (IF)	Consequent (THEN)	Defuzzification
Mamdani [19]	Fuzzy sets (scalar, parameterized)	Fuzzy sets (scalar, parameterized)	Center of gravity
TS [20]		Functional (often linear)	Fuzzily weighted sum
ANYA [26]	Data *clouds* (non-parametric)	Any of the above two types	

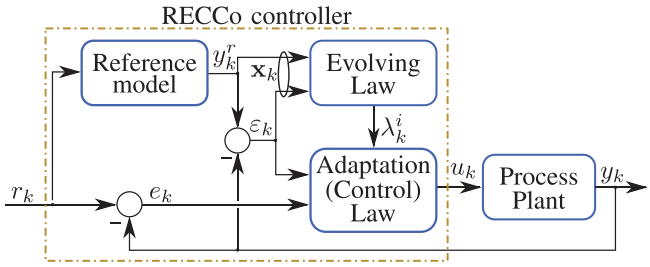

Fig. 1. Control scheme of the RECCo algorithm.

The remainder of this paper is organized as follows. In Section 2 the RECCo algorithm is presented, including the evolving structure and the adaptation law of the controller. The normalized data (cloud) space is explained in Section 3 and moreover, several experiments are provided to prove the benefit of the proposed approach. In Section 4 the performance comparison between RECCo and PID controller is presented. Moreover, the PHE process is explained and the experiment is provided to show the performance and the ability of learning of the RECCo controller in practice. At the end in Section 5 the conclusions are given.

2. Robust Evolving Cloud-based Controller (RECCo)

2.1. The structure of the RECCo controller

In this section the RECCo controller will be described. The control algorithm consists of three different parts: reference model, evolving law, and adaptation law. All this parts are schematically presented in Fig. 1. Theoretically, the controller could be initialized from the first data sample received. But of course, any existing information about the controlled process can be used to suitably initialize the design parameters. After the initialization, for every incoming sample the controller gains are adapted and, if the certain conditions are satisfied, a new data cloud (fuzzy rule) is added.

The Robust Evolving Cloud-based Controller (RECCo) is a type of ANYA fuzzy rule-based system with non-parametric antecedents (IF part). As we mentioned above, this method applies the concept of fuzzy data clouds and normalized relative data density to define the membership of the current data[1] to the existing clouds. The clouds represent sets of previous data samples which are close to each other. Incoming data samples are analyzed in an online manner and each sample is associated with one of the clouds and only the parameters of that cloud are updated.

As we already said, the RECCo controller is based on the ANYA FRB system proposed in [26] and has the following form:

$$\mathcal{R}^i : \text{IF} \quad (\mathbf{x} \sim X^i) \quad \text{THEN} \quad (u^i) \tag{1}$$

where the number of rules $\mathcal{R}^i$ is equal to the number of the clouds in the data space $i = 1, \ldots, c$, and moreover, it changes during the control process. The non-parametric antecedent part is defined with the operator $\sim$ which could be linguistically expressed as '*is associated with*' and that means that the current data $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is related to the ith cloud $X^i \in \mathbb{R}^n$. The consequent part is defined by c different (partial) control actions u^i for each rule. RECCo controller can work with different forms of defuzzification such as weighted average, center-of-gravity, "winers takes all" and some mixed forms (e.g. parameterized defuzzification).

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

$$\lambda_k^i = \frac{\gamma_k^i}{\sum_{j=1}^{c} \gamma_k^j}, \quad i = 1, \ldots, c \tag{2}$$

where γ_k^i is the local density of the ith cloud for the current data $\mathbf{x}$. The local density calculation in the following subsection will be explained in detail, together with the evolving law of the RECCo controller.

2.2. The procedure of the RECCo control algorithm

2.2.1. Reference model

Choosing an appropriate reference model is a very important part of the proposed adaptive system design. General suggestions for selecting the reference model dynamics are that the time constants have to be similar (usually slightly shorter) to those of uncontrolled process. The reference model order would be at less or equal to the order of the plant [35]. Furthermore, the initial conditions of the reference model would then need to have the same values as the initial plant ($y_0^r = y_0$).

The reference model part of the RECCo controller defines the desired trajectory y_k^r and the dynamics that the plant output y_k should follow. In this case we define simple first order linear reference-model as:

$$y_{k+1}^r = a_r y_k^r + (1 - a_r) r_k \quad 0 < a_r < 1 \tag{3}$$

where the parameter a_r is the pole of that model. It can be approximated by $(1 - T_s/\tau)$, where T_s is the sampling period of the process and τ is the time constant of the reference model which is slightly shorter than the estimated time constant of the controlled plant. In (3) the r_k is the reference signal and the y_k^r represents the desired trajectory of the plant output y_k. The goal of the controller, is to provide efficient performance and to ensure that the tracking error:

$$\varepsilon_k = y_k^r - y_k \tag{4}$$

is as small as possible (in the presence of disturbances and modeling errors).

When dealing with adaptive and evolving (online learning) systems we need to construct reference with changing steps in some operating range $[r_{min}, r_{amx}]$. In this case the user (operator) of the process only chooses the limit values r_{min} and r_{max} and the RECCo algorithm constructs the step changes in this interval.

We have to note here that the RECCo controller is not limited only to this type of reference model (first order linear model), but also other types could be used according to the dynamics of the controlled process.

2.2.2. Evolving law

The evolving law in this paper consists only a mechanism for adding new clouds (rules). Beside this, another evolving mechanisms such as merging, splitting and removing clouds can be also implemented. We decide to use just adding mechanism due to simplicity of the implementation and because it is sufficient for control the plant proposed in Section 4. The adding mechanism relies

[1] Data will be used to express singular and plural form in this paper.

on the local density γ_k^i of the current data sample with the existing clouds. The local density takes into consideration **all** the data samples from one particular cloud (therefore local) and is calculated using a suitable kernel $\mathcal{K}$:

$$\gamma_k^i = \mathcal{K}\left(\sum_{j=1}^{M^i} d_{kj}^i\right) \tag{5}$$

where M^i is the number of data samples in ith cloud and d_{kj}^i is the distance between the current data sample $\mathbf{x}_k$ and the jth sample $\mathbf{x}_j^i$ from the ith cloud. In all the equations the superscript in variables (e.g. i in $\mathbf{x}_k^i$) refers to the clouds, while the subscript refers to the time stamp (e.g. k in $\mathbf{x}_k^i$). As we can see in (5) this approach directly takes into account all previous data samples.

In this article we used a Cauchy kernel as was proposed in [26] and the local density of the ith cloud is defined as follows:

$$\gamma_k^i = \frac{1}{1 + \frac{\sum_{j=1}^{M^i}(d_{kj}^i)^2}{M^i}} \tag{6}$$

where $\sum_{j=1}^{M^i}(d_{kj}^i)^2$ is the sum of the square of Euclidean distances ($d_{kj}^i = \|\mathbf{x}_k - \mathbf{x}_j^i\|^2$) between the new data $\mathbf{x}_k$ and all data points of the ith cloud. We have to mention that, another type of distance measure could also be used (e.g. Mahalanobis in [30]) and it was shown that both Euclidean and Mahalanobis distance produced satisfying results. For easier practical and computational implementation, local density (6) can be recursively rewritten as follows:

$$\gamma_k^i = \frac{1}{1 + \|\mathbf{x}_k - \mu_k^i\|^2 + \sigma_k^i - \|\mu_k^i\|^2} \tag{7}$$

where μ_k^i is the mean value of the cloud's data points and σ_k^i is the mean-square length of the data vectors in the ith cloud. Both of them can be recursively calculated using following equations for mean value and mean-square length, respectively:

$$\mu_k^i = \frac{M^i - 1}{M^i}\mu_{k-1}^i + \frac{1}{M^i}\mathbf{x}_k \tag{8}$$

$$\sigma_k^i = \frac{M^i - 1}{M^i}\sigma_{k-1}^i + \frac{1}{M^i}\|\mathbf{x}_k\|^2 \tag{9}$$

Initial condition ($M^i = 1$) for the mean value is $\mu_1^i = \mathbf{x}_1$ and for the mean-square length is $\sigma_1^i = \|\mathbf{x}_1\|^2$.

The evolving law in this paper consists the mechanism of adding new clouds and is the same as the one presented in [28]. Moreover, it is much simpler in comparison to the mechanism used in [27,29]. Once a new data sample arrive we need to calculate c different local densities between the sample and all the existing clouds (see Fig. 2). According to the maximal local density ($\max_i \gamma_k^i$) the data sample is associated with that cloud and furthermore, the parameters of that cloud are updated using Eqs. (8) and (9). Theoretically, it is possible to happen that the current data sample has the same density to two or more clouds. In that case we associate that data sample with the oldest cloud (the one that was added before the others). But, if the maximal local density ($\max_i \gamma_k^i$) is lower than the threshold value γ_{max} (the current data sample is far away from all existing clouds), a new cloud is added. The cloud's data space is normalized (it will be explained in the next section) and due to this the default value of the threshold can be fixed $\gamma_{max} = 0.93$. Some conservatism is always welcome when changing the structure of the evolving system. This is why some other criteria need to be fulfilled before adding a new cloud (such as certain time n_{add} has passed from the last change). We have to note here that in our previous and current experiments we always use default value of this parameter $n_{add} = 20$. Moreover,

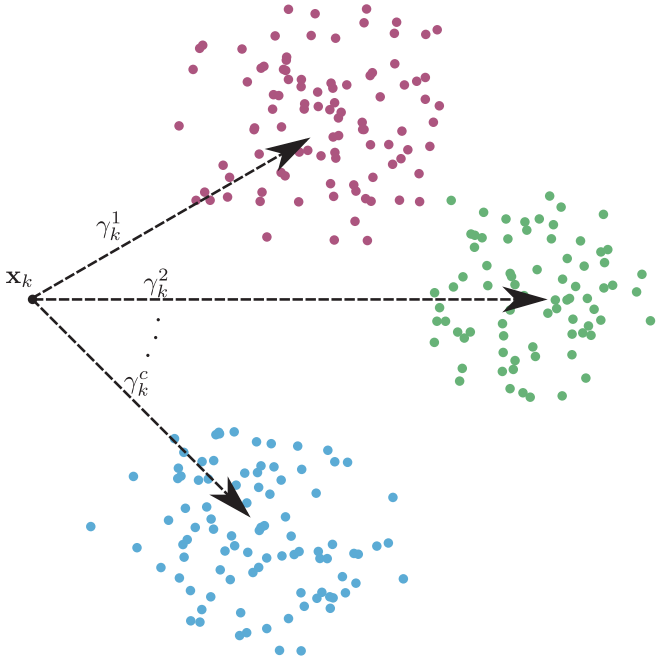

Fig. 2. Associating the current data sample $\mathbf{x}_k$ with one of the existing clouds according to the local densities γ_k^i, where $i = 1, \ldots, c$.

because of the normalized data space and fixed value of the parameter γ_{max} the adding of new clouds is more stable and the parameter n_{add} can be even neglected. We can summarize the whole evolving procedure presented above in the pseudo Algorithm 1 (see lines from 9 to 22).

Algorithm 1. Pseudo code of the RECCo PID control algorithm.

1: Initialize (Process parameters): τ, T_s, u_{min}, u_{max}, r_{min}, r_{max}.
2: Initialize (Evolving parameters): γ_{max}, $c = 0$, c_{max}, n_{add}.
3: Initialize (Adaptation parameters): α_P, α_I, α_D, α_R, σ_L, d_{dead}, $\underline{\theta}$, $\overline{\theta}$.
4: **repeat**
5: Measurement: y_k.
6: Define and compute: y_k^r ▷ *Reference model*
7: Compute: e_k, ε_k, Σ_k^{ε}, Δ_k^{ε}.
8: Compute: $\mathbf{x}_k = [\ \varepsilon_k/\Delta\varepsilon,\ (y_k^r - r_{min})/\Delta r\ ]^T$.
9: **if** $c = 0$ **then** ▷ *Start of the evolving law*
10: Increment: c,
11: Store: k_{add},
12: Initialize: μ_0^1, σ_0^1, θ_0^1.
13: **else**
14: Calculate: γ_k^i, λ_k^i, where $i = 1, \ldots, c$
15: **if** $(max_i\gamma_k^i < \gamma_{max}$ **and** $k > (k_{add} + n_{add}))$ **then**
16: Increment: c,
17: Store: k_{add},
18: Initialize: μ_0^c, σ_0^c, θ_0^c.
19: **else**
20: Associate sample $\mathbf{x}_k$ with cloud $(max_i\gamma_k^i)$
21: Update μ_k^i, σ_k^i for the cloud $(max_i\gamma_k^i)$
22: **end if**
23: **end if** ▷ *End of the evolving law*
24: Adaptation of the **PID controller gains**.
25: Computation of the **control law**.
26: **until** End of data stream.

2.2.3. Adaptation law

For the consequent part of the RECCo controller the PID-type control is used [28] and each cloud (fuzzy rule) has its own PID parameters. The vector of the parameters is denoted as $\theta_k^i = [P_k^i, I_k^i, D_k^i, R_k^i]^T$ and parameters of the first cloud are initialized with zeros $\theta_0^1 = [0, 0, 0, 0]^T$, while all later added clouds are initialized with mean value of the parameters of all previous clouds as follows:

$$\theta_0^c = \frac{1}{c-1}\sum_{j=1}^{c-1}\theta_k^j \tag{10}$$

where c is the index of the newly added cloud.

After the classification of the current data sample to one of the clouds, only the PID parameters of that cloud are adapted while the parameters of other clouds are kept constant:

$$\theta_k^i = \theta_{k-1}^i + \Delta\theta_k^i \tag{11}$$

and the adaptation of the PID parameters was introduced in [28], but in this article we proposed an improved version as follows:

$$\begin{aligned}
\Delta P_k^i &= \alpha_P\, G_{sign}\lambda_k^i \frac{|e_k\varepsilon_k|}{1+r_k^2}\\
\Delta I_k^i &= \alpha_I\, G_{sign}\lambda_k^i \frac{|e_k\Delta_k^\varepsilon|}{1+r_k^2}\\
\Delta D_k^i &= \alpha_D\, G_{sign}\lambda_k^i \frac{|e_k\Delta_k^\varepsilon|}{1+r_k^2}\\
\Delta R_k^i &= \alpha_R\, G_{sign}\lambda_k^i \frac{\varepsilon_k}{1+r_k^2}
\end{aligned} \tag{12}$$

where α_P, α_I, α_D, α_R are the adaptation gains of the controller parameters, $G_{sign} = \pm 1$ is the known process gain sign, λ_k^i is the normalized local density of the cloud, ε_k is the tracking error while the control error is denoted as $e_k = r_k - y_k$. The discrete-time derivative is denoted as Δ_k^ε and will be discussed later. In (12) only the adaptation gains should be set initially. The default value of the parameters is 0.1 and is used when the range of the control variable is ($u_{min} = 0/4$, $u_{max} = 20$). When the range is different, the value of the parameters is scaled as follows:

$$\alpha_{new} = \frac{u_{\max} - u_{\min}}{20}\cdot 0.1$$

For example if the range is from $u_{min} = 0$ to $u_{max} = 100$ the new value of the adaptive gains will be $\alpha_{new} = 0.5$.

The absolute values in (12) are used only in the starting phase of the control performance (five time constants is enough) and after that they are omitted from the adaptation law. The problem appears when the initial value of the controlled variable is higher than the reference value ($y_0 > r_0$) which causes negative control error ($e_k < 0$) and correspondingly negative adaptation of the parameters ($\Delta P_k^i, \Delta I_k^i, \Delta D_k^i < 0$). As we already mentioned, the parameters of the first data cloud are initialized with zeros and without the absolute values in (12) the negative adaptation will lead to even bigger error. Fig. 3 shows the difference in the control performance with and without using absolute values in the starting phase. On the other hand, this adaptation law does not alter the performance of the controller in the case of the positive initial error, moreover, does not depend on the sign G_{sign} of the process gain.

In Section 1 we noted that the proposed approach can work with both TS and Mamdani (Table 1) rule consequent part. For a PID-type controller, the rule consequent has the following form:

$$u_k^i = P_k^i\varepsilon_k + I_k^i\Sigma_k^\varepsilon + D_k^i\Delta_k^\varepsilon + R_k^i, \quad i = 1, \ldots, c \tag{13}$$

where P_k^i, I_k^i, D_k^i are controller gains while R_k^i is compensation of the operating point. While the adaptation of the parameter R_k^i in (12) is driven only by tracking error ε_k this parameter tries to correct the offset error of the current operating point. Σ_k^ε and Δ_k^ε in (13) are discrete-time integral and derivative of the tracking error, respectively, and can be calculated as follows:

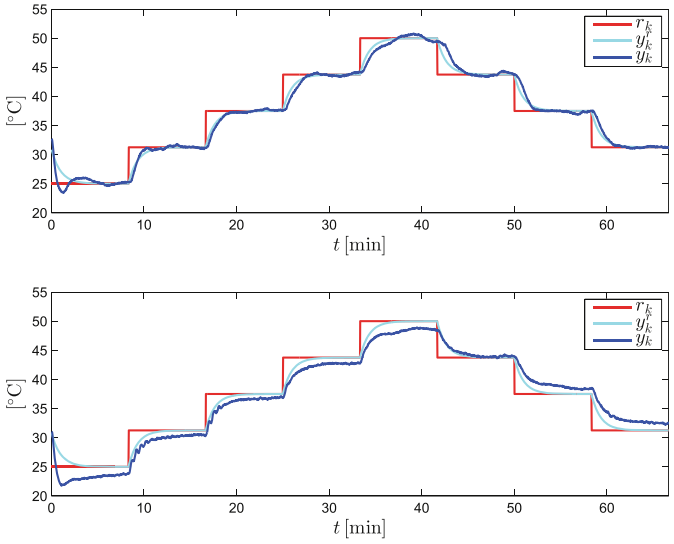

Fig. 3. The reference, the model reference and the controlled signal for heat-exchanger pilot plant in the starting phase with (upper plot) and without (lower plot) calculating the absolute value in adaptation law of the controller gains ($y_0 = 32\,^\circ$C, $r_0 = 25\,^\circ$C).

$$\Sigma_k^\varepsilon = \sum_{\kappa=0}^{k-1}\varepsilon_\kappa = \Sigma_{k-1}^\varepsilon + \varepsilon_{k-1} \tag{14}$$

$$\Delta_k^\varepsilon = \varepsilon_k - \varepsilon_{k-1} \tag{15}$$

Finally, for the defuzzification the weighted average is used (but not limited to this form) and furthermore, the control variable becomes:

$$u_k = u_{\min} + \sum_{i=1}^{c}\lambda_k^i u^i = u_{\min} + \frac{\sum_{i=1}^{c}\gamma_k^i u^i}{\sum_{i=1}^{c}\gamma_k^i} \tag{16}$$

where u^i denotes the ith (partial) rule consequent and normalized relative density (2) is used. From the practical implementation point of view we add u_{min} in this equation (in comparison with the one proposed in [27,28]) and represents the minimal input value of the real actuator which in our case is $u_{min} = 4$ mA.

2.3. The instability protection mechanism

This section is devoted to the modifications of the adaptation law (11) that improve the robustness of the closed-loop system. Supervised adaptation of any controller can improve, theoretically and practically, the performance and robustness of the controller. In order to minimize the negative influence of parasitics, disturbances in the system and to eliminate the pure integral action of the adaptive law, we introduce several mechanisms to improve the RECCo control algorithm.

When dealing with adaptive controllers and parameter adaptation we need to be aware of the potential instability problems caused by the parameter drift [36]. Due to this, to make RECCo controller more robust, several techniques were already applied in [27,28]. In this paper we will use the following techniques:

2.3.1. Dead zone in the adaptation law

To improve the robustness under the unknown bounded disturbances and modeling errors, the RECCo controller includes a

dead-zone in adaptation law. The general idea behind the dead-zone mechanism, in case of bounded disturbances, is to turn off the adaptation algorithm when the absolute value of the tracking error is smaller than a certain threshold [37]:

$$\Delta\bar{\theta}_k^i = \begin{cases} \Delta\theta_k^i & |\varepsilon_k| \geq d_{dead} \\ 0 & |\varepsilon_k| < d_{dead} \end{cases} \quad i = 1, \ldots, c \tag{17}$$

The parameter d_{dead} should be chosen slightly larger than the process noise to improve the effectiveness of the adaptive law. A larger threshold implies a shorter adaptation period and larger tracking error, while smaller value can lead to parameter drift.

2.3.2. Parameter projection

Parameter projection mechanism is used to guarantee that the estimation of the parameters will stay within finite known region [38]. In the case of the positive plant gain all the parameters should be bounded by 0 from bellow while upper bound may or may not be provided. The adaptive law in (11) is generalized as follows:

$$\theta_k^i = \begin{cases} \theta_{k-1}^i + \Delta\theta_k^i & \underline{\theta} \leq \theta_{k-1}^i + \Delta\theta_k^i \leq \bar{\theta} \\ \underline{\theta} & \theta_{k-1}^i + \Delta\theta_k^i < \underline{\theta} \\ \bar{\theta} & \theta_{k-1}^i + \Delta\theta_k^i > \bar{\theta} \end{cases} \quad i = 1, \ldots, c \tag{18}$$

In our case we chose $\underline{\theta} = 0$ and $\bar{\theta} = \infty$ for the controller gains P_k, I_k, and D_k, while for the compensation of the operating point R_k the lower bound was $\underline{\theta} = -\infty$. If we have some a priori knowledge where the true parameters θ^* are located in $\mathcal{R}^n$ we can define upper and lower bound for the elements of θ. The benefit of such information may speed up the convergence of finding optimal parameters.

2.3.3. Leakage in the adaptation law

The use of leakage in the adaptation law is a very known approach for improvement of robustness of adaptive control. Already exist different types of leakage, for example σ-modification [39], e_1-modification [40], switching σ-modification [41], etc.

Including the leakage in the adaptation law results in:

$$\theta_k^i = (1 - \sigma_L)\theta_{k-1}^i + \Delta\theta_k^i \quad i = 1, \ldots, c \tag{19}$$

where σ_L defines the extent of the leakage.

2.3.4. Interruption of adaptation

In the RECCo algorithm we first calculate the adaptation of the PID parameters ($\Delta\theta_k^i$) and then the control variable u_k. In some cases these two steps can be in conflict, which means that the adaptation causes control signal which is outside the limits $[u_{min}, u_{max}]$. In such case the adaptive law should be interrupted in the following manner:

$$\Delta\bar{\theta}_k^i = \begin{cases} \Delta\theta_k^i & u_{\min} \leq u_k \leq u_{\max} \\ 0 & \text{otherwise} \end{cases} \quad i = 1, \ldots, c \tag{20}$$

3. Cloud space normalization

Until now, we did not discuss the content and the definition of the data sample $\mathbf{x}_k$. In the previous work [28] the data sample was defined in 2D space as $\mathbf{x}_k = [\varepsilon_k, y_k^r]^T$, where the first element (ε_k) represents the horizontal axis while the second element (y_k^r) represents the vertical axis of the data space. In this case, if we want to change the operating range of the reference signal r_k, this will also affect the reference model output y_k^r, and consequently the data space will change its size (shrink or expand) in both direction (y_k^r and ε_k).

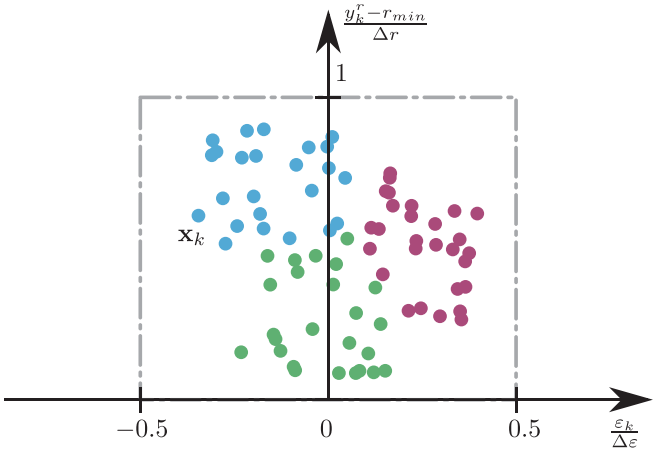

Fig. 4. Normalized cloud space.

Our idea is to define a constant data space (see Fig. 4), where majority of the data will appear, regardless of the range of the reference signal. Even if we want to control a different process, the same data normalization can be used with the same constant data space. As a consequence, the evolving parameter γ_{max} can be fixed. We propose a normalized data space as follows:

$$\mathbf{x} = \left[\frac{\varepsilon_k}{\Delta\varepsilon}, \frac{y_k^r - r_{\min}}{\Delta r} \right]^T \tag{21}$$

where $\Delta r = r_{max} - r_{min}$ and $\Delta\varepsilon = \frac{\Delta r}{2}$. In this case the operator (user) needs to choose, according to the process requirements, only the operating range of the plant $[r_{min}, r_{max}]$. After that, several step changes of the reference signal r_k are constructed to cover the whole range of the process (e.g. see upper plot in Fig. 12).

4. Experimental results of a heat-exchanger pilot plant

4.1. Comparison between RECCo and classical PID controllers

In this section, the performance of the RECCo controller in comparison with the classical PID controller is studied. Three methods for designing the parameters of the PID controller were used: Ziegler–Nichols [32] (PID_{ZN}), Cohen–Coon [33] (PID_{CC}) and pole placement method [34] (PID_{PP}). To verify the performance of the proposed control algorithm a model of plate heat-exchanger (PHE) was used [42]. The control procedure of the RECCo controller for PHE model is described in [43]. The same procedure for acquiring the parameters and the structure of the RECCo controller in this paper was used.

From the open loop response of the PHE model (see Fig. 5), the characteristic parameters such as process gain K_P, time constants τ_P and dead time $T_{dead,P}$ were obtained. Due to the non-linearity of the process model, each operating point has different characteristic parameters (see Table 2). An exception is the dead time which is constant. For the gain and the time constant of the process we calculate average value. Therefore, the average process gain is $K_P = 2.49$ while the average time constant is $\tau_P = 35.25$. These parameters were used to determine the values of the PID controllers (PID_{ZN}, PID_{CC}, and PID_{PP}). The sampling time used is $T_s = 2$s.

For the performance evaluation, several criteria functions were compared, such as maximum overshoot, rising and settling time. Furthermore, the control effort was evaluated using integral criteria

Table 2
Simulation. Process characteristics of the PHE model: Process gain K_P, time constant τ, and dead time t_{dead}.

u_k [mA]	6	8	10	12	14	16	18	20
K_P	3.38	4.17	4.93	2.71	1.21	1.73	1.10	0.68
τ_P [s]	48	50	27	19	18	47	44	29
$T_{dead,P}$ [s]	4	4	4	4	4	4	4	4

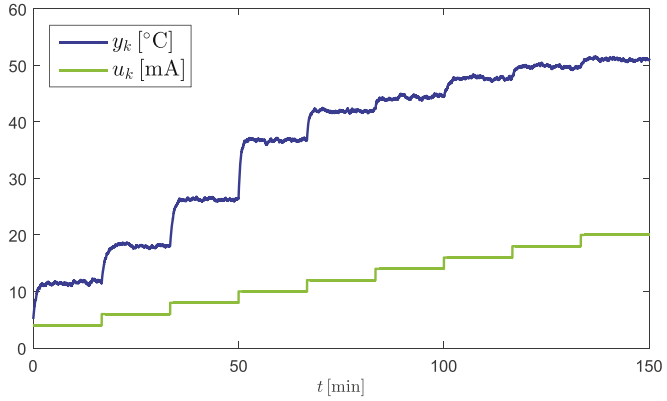

Fig. 5. Simulation. Open-loop response of the plate heat-exchanger model.

functions: Sum of the Absolute Input differences (f_{SAdU}) and Sum of the Squared Input differences (f_{SSdU}):

$$f_{SAdU} = \sum_k |\Delta u_k| \tag{22}$$

$$f_{SSdU} = \sum_k \Delta u_k^2 \tag{23}$$

where $\Delta u_k = u_k - u_{k-1}$ is a change of the input action.

Besides evaluating the control effort, we also compared integral (cumulative) of the process error ($e_k = r_k - y_k$) using the following criteria functions:

$$f_{SAE} = T_s \sum_k |e_k| \tag{24}$$

$$f_{SSE} = T_s \sum_k |e_k^2| \tag{25}$$

where T_s is the sampling time and in our case is equal to 2.

Finally, the comparison results are presented in the next figures and tables. First, in Fig. 6 the controlled variables obtained by each of the controllers (PID_{ZN}, PID_{CC}, PID_{PP} and $RECCo$) are presented.

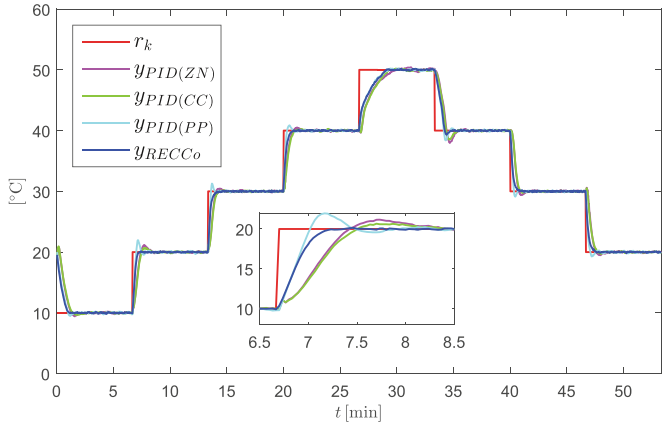

Fig. 6. Simulation. The comparison of the responses (controlled variables) obtained by different controllers.

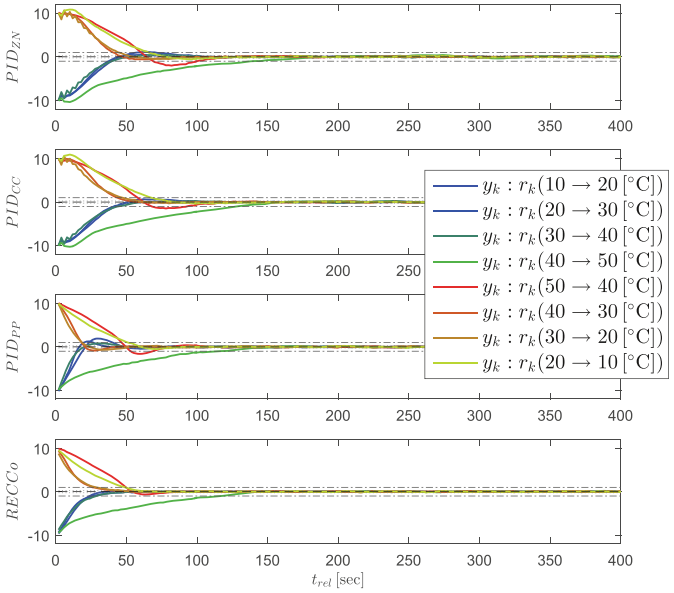

Fig. 7. Simulation. Each of the four plots shows the performance of a controller in 8 different transients of the controlled variable (all the transients are appropriately shifted).

For better comparison, the same controlled variables form Fig. 6 are also shown in Fig. 7 where the different operating points are plotted in the same time frame. The rise time (the time required by the response y_k to rise from 10% to 90% of its final value), maximal overshoot and settling time (e_k is smaller than 0.25 °C) were calculated and the results are presented in Table 3. Focusing on the rise time from the table, in some cases the pole placement controller provide better results, but the difference is in range of few time samples. Comparing the maximal overshoot and the settling time the RECCo controller provides better results.

Table 4 summarizes the comparison between the four controllers based on four criteria (22)–(25). We can notice that for

Table 3
Simulation. Performance comparison between the four controllers for plate heat-exchanger.

r_k [°C]	10	20	30	40	50	40	30	20
Rise time [s]								
PID_{ZN}	44	36	34	36	122	42	26	42
PID_{CC}	46	40	38	42	112	42	28	44
PID_{PP}	46	14	12	12	110	38	14	16
$RECCo$	42	20	24	28	104	36	22	28
Overshoot [%]								
PID_{ZN}	5.8	11.7	4.5	5.0	4.2	20.1	5.9	5.1
PID_{CC}	3.5	6.6	1.6	2.9	2.2	14.8	2.5	3.0
PID_{PP}	5.1	19.6	12.7	8.6	2.3	16.0	8.6	7.1
$RECCo$	1.9	1.4	1.6	1.3	2.0	6.8	1.3	2.0
Settling time [s]								
PID_{ZN}	116	98	78	94	382	168	78	100
PID_{CC}	122	96	54	242	154	114	68	82
PID_{PP}	150	238	34	48	144	102	38	74
$RECCo$	60	30	42	42	138	80	42	48

Table 4
Simulation. Comparison between the four controllers using different criteria functions.

	f_{SAdU} [A]	f_{SSdU} [A^2]	f_{SAE} [°C]	f_{SSE} [°C^2]
PID_{ZN}	2.93	18.52	1419	25,430
PID_{CC}	2.87	18.64	1121	20,277
PID_{PP}	1.77	9.28	327	6110
RECCo	1.59	6.48	309	5129

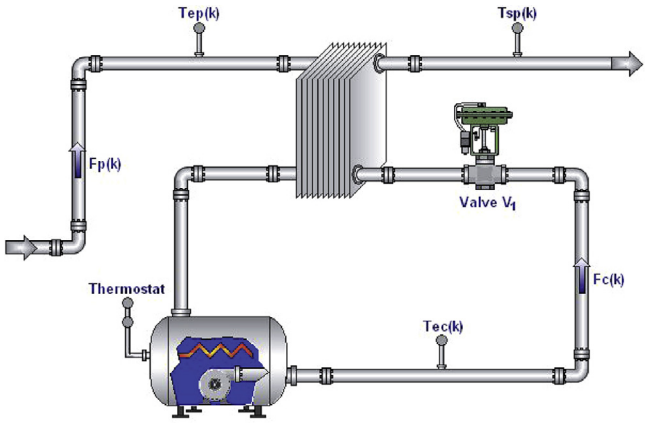

Fig. 8. Plate heat-exchanger pilot plant process.

all criteria functions the RECCo controller indicates better performance in controlling the PHE process.

4.2. Real system

In our experimental study a real plant of plate heat-exchanger (PHE) is used. The main purpose of this device is to efficiently transfer the heat from one medium to another. In Fig. 8 the process scheme is shown. It consists of two separate water circuits (the primary one is hot flow and the secondary is cold water flow). The primary circuit has a constant inlet temperature $T_{ec}(k)$ controlled by on-off thermostat which characteristic will be discussed later. Motor driven valve V_1 controls the primary circuit flow $F_c(k)$ and represents the control variable u_k. The outlet water of the plate exchanger of the primary circuit is returned to the reservoir. The secondary circuit has inlet temperature $T_{ep}(k)$ and the constant water flow of cold water $F_p(k)$ on one side and the controlled variable is outlet temperature $T_{sp}(k)$ on the other side of the circuit.

The practical implementation of the whole system (RECCo control algorithm, data acquisition system and PHE plant) for process control is shown in Fig. 9. We can notice that RECCo algorithm requires only two connections to the real process, u_k and y_k, without additional information of the process. The control signal u_k is in the range 4–20 mA and is not additionally converted because both, the RECCo algorithm and the PHE process work in the same range. On the other hand, the temperature sensor provides the signal in the range 4–20 mA and additionally, we need to convert this signal to the actual temperature range of the sensor (from 0 °C to 100 °C) for easier interpretation and understanding of the results.

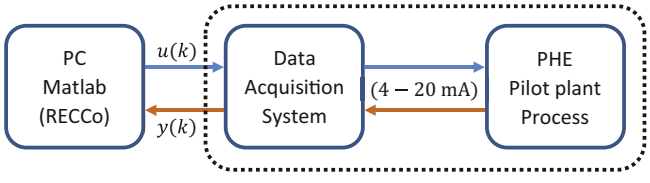

Fig. 9. Process control and data acquisition system.

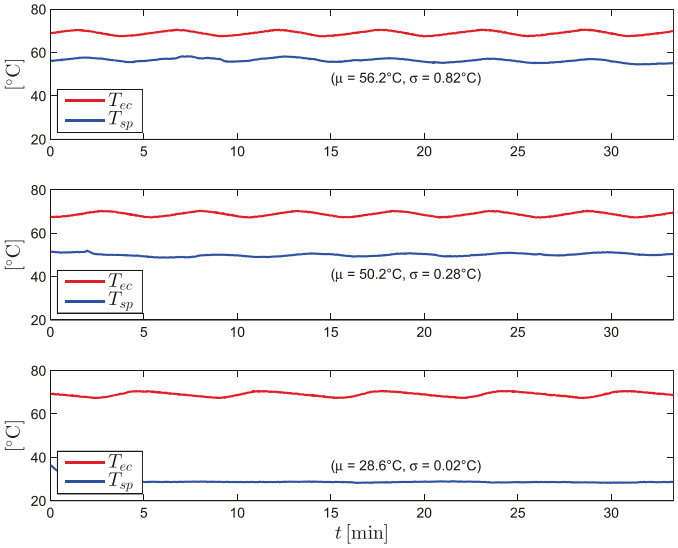

Fig. 10. The variation of the inlet temperature T_{ec} due to the primary circuit thermostat at constant valve opening in three different operating points (top u_k = 20 mA, middle u_k = 13.3 mA and bottom u_k = 4.6 mA).

The wide hysteresis of the thermostat (approximately ±2 °C) in the primary circuit represents the disturbance and has different influence across the operating region. This could be seen in Fig. 10 where a constant valve command is applied in three operating points. The variances and the mean values of the signals $T_{sp}(k)$ for each operating point are then calculated. The changing influence of the thermostat characteristics in different operating points provides an additional complexity to the process and makes the control problem more challenging.

The open loop response of the plant is shown in Fig. 11. The step changes of the input signal are chosen to cover the whole range of the process (from 4 mA to 20 mA in steps of 1 mA). It can easily be noticed the nonlinearity of the process: changing time constant depending on the operating point, and also the effect of the thermostat hysteresis to the process. As already said, the effect of the thermostat is larger in the higher operating points. All these

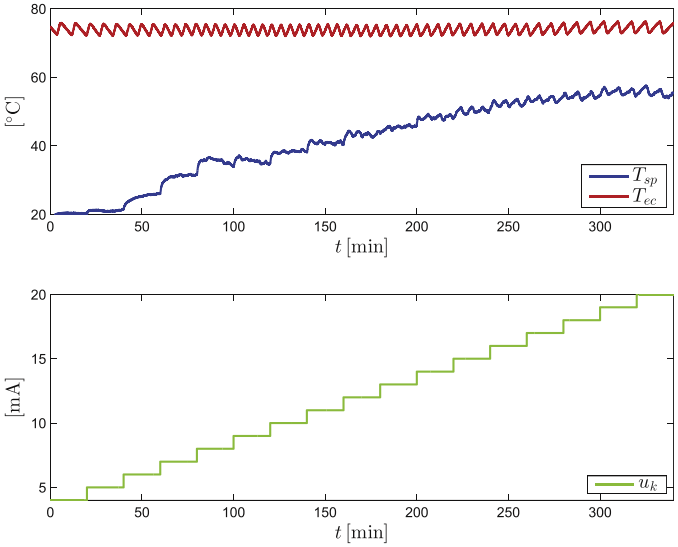

Fig. 11. Open-loop response of the PHE output T_{sp} to the series of step changes on the input valve V_1 (u_k changes from 4 mA to 20 mA in steps of 1 mA, bottom plot). Top plot shows the inlet water temperature T_{ec} (red line) and the outlet water temperature T_{sp} (blue line). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

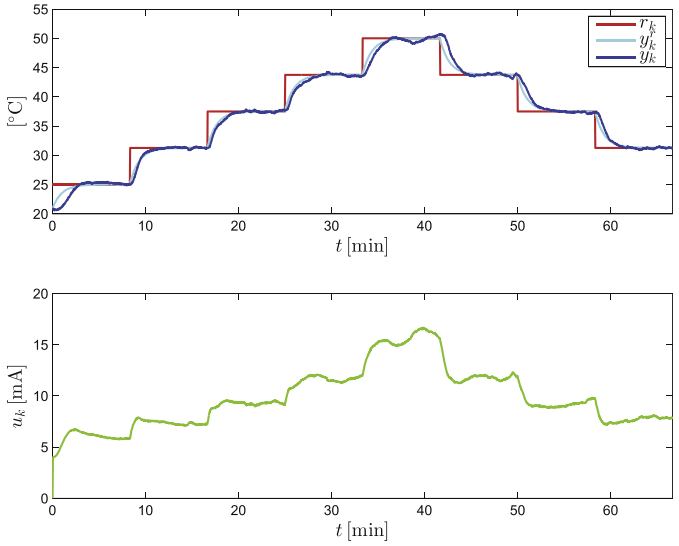

Fig. 12. The reference, the model reference and the controlled signal (top plot) and the control signal (bottom plot) for plate heat exchanger in the starting phase.

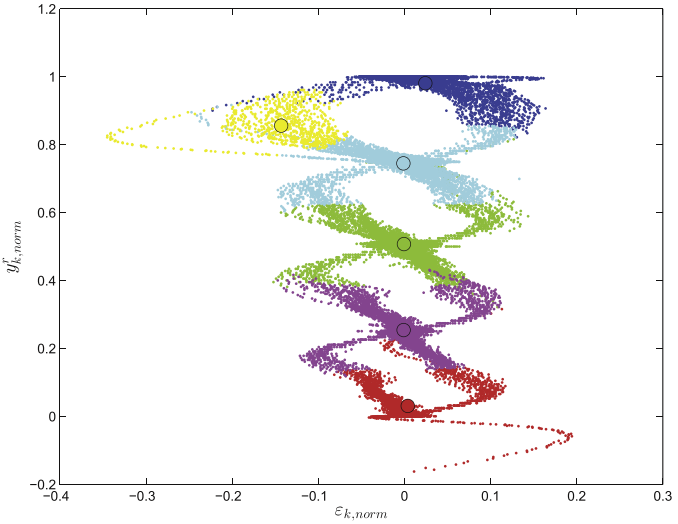

Fig. 14. The clouds in the input space $\mathbf{x} = \left[\frac{\varepsilon_k}{\Delta\varepsilon}, \frac{y_k^r - r_{\min}}{\Delta r}\right]^T$.

characteristics of the PHE process are dealt with RECCo controller and the results of the proposed algorithm are shown in the following.

Advantage of the RECCo controller is that we need only very basic information of the controlled process (estimated value of the dominant time constant τ, the range of the actuator $[u_{min}, u_{max}]$, and the range of the controlled variable $[r_{min}, r_{max}]$). Furthermore, the controller's structure is evolved and parameters are adapted during performing the control of the process. The design parameters are divided into three groups (process, evolving and adaptive parameters), the same as in the initialization phase of Algorithm 1.

(1) The first group contains very technical (process) parameters which are set according to the system requirements. The process range in this experiment was chosen as $r_{min} = 25\,^\circ$C and $r_{max} = 50\,^\circ$C. We mentioned above that in the case of a different process range, no additional tuning of adaptive and evolving parameters is required. The time constant and the sampling time of the reference model were chosen as $\tau = 40$ s and $T_s = 2$ s, respectively. The time constant τ is chosen in the range of the time constant of the plant. The process input or the actuator's range was defined by the hardware ($u_{min} = 4$ mA, $u_{max} = 20$ mA).

(2) The evolving parameters from the second group define the rules when and why a new cloud is added. Simulations were started with zero fuzzy clouds (rules). A new cloud is added when the maximal value of the local densities $\gamma_k^i, i = 1, \ldots, c$ is lower than the threshold value $\gamma_{max} = 0.93$. The parameter that defines the minimum number of samples between two new clouds is defined as $n_{add} = 20$.

(3) The third group contains parameters of the adaptation/control law. The dead zone d_{dead} was chosen as 1% of the process range Δr ($d_{dead} = 0.25\,^\circ$C) and the leakage parameter is set to $\sigma_L = 10^{-6}$. All adaptive gains α_P, α_I, α_D and α_R are set to 0.1.

In this section the practical results are presented and the improvements of the RECCo controller (adaptation of the parameters and cloud space normalization) are tested. A closer look to the starting phase of the experiment is shown in Fig. 12 where the reference r_k, the model reference output y_k^r, the controlled signal y_r, and the control signal u_k are given. At this point we notice that

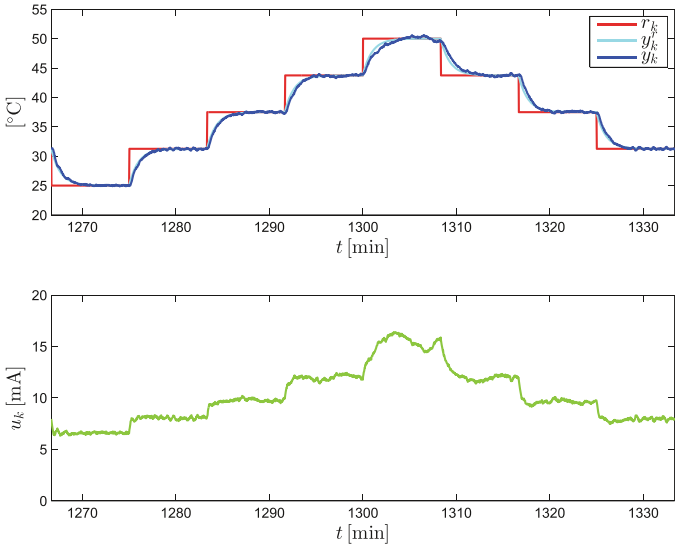

Fig. 13. The reference, the model reference and the controlled signal (top plot) and the control signal (bottom plot) for plate heat exchanger in the finishing phase.

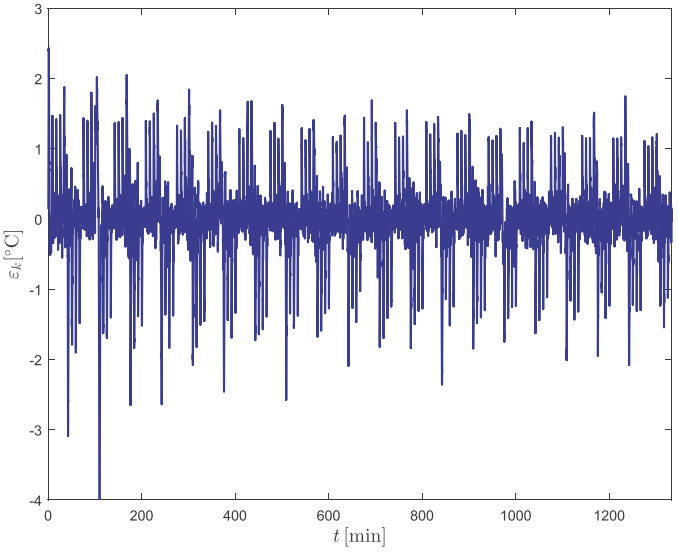

Fig. 15. The tracking error ε_k for heat-exchanger pilot plant ($y_0 = 15\,^\circ$C).

in this experiment a new adaptation method is used (mentioned in Section 2.2.3). In Fig. 13 the phase after the transient of the adaption is shown. In Fig. 14 all added clouds during the process control are shown. In this case six clouds (fuzzy rules) have been constructed. The tracking error ε_k is shown in Fig. 15 where its decreasing with time can be clearly noticed.

5. Conclusion

In this paper a practical implementation of the RECCo control approach on a real heat-exchanger was proposed. Moreover, a new approach of RECCo with normalized data space and improved adaptation of controller parameters was used. The normalization of the data space results in making the problem of determination of the evolving and the adaptation parameters much easier. In fact, default choice of design parameters provides very good control performance for a broad specter of processes as shown in our numerous studies. In case of the negative initial control error a new adaptation of the controller is proposed for the starting phase of the evolving process. Both modifications, normalization and adaptation, are thoroughly tested and analyzed for real plate heat exchanger plant. Irrespective of the process range, the same initial values of the parameters are used in all the experiments which is one of the most valuable benefits of the proposed modifications. The effect of the input disturbances was also analyzed to test the robustness of the controller. The main advantage of the RECCo controller is the self-evolving procedure which starts with a very limited a priori information about the control process (only the range of the control and controlled variable are needed and a rough estimate of the dominant time constant of the controlled process). This approach effectively deals with nonlinear processes.

References

[1] W.-D. Chang, R.-C. Hwang, J.-G. Hsieh, A self-tuning PID control for a class of nonlinear systems based on the Lyapunov approach, J. Process Control 12 (2002) 233–242.
[2] Y. Kansha, L. Jia, M.-S. Chiu, Self-tuning PID controllers based on the Lyapunov approach, Chem. Eng. Sci. 63 (10) (2008) 2732–2740.
[3] C. Riverol, V. Napolitano, Use of neural networks as a tuning method for an adaptive PID: application in a heat exchanger, Chem. Eng. Res. Des. 78 (8) (2000) 1115–1119.
[4] A. Altinten, S. Erdogan, F. Alioglu, H. Hapoglu, M. Alpbaz, Application of adaptive PID control with genetic algorithm to a polymerization reactor, Chem. Eng. Commun. 191 (9) (2004) 1158–1172.
[5] A. Moharam, M.A. El-Hosseini, H.A. Ali, Design of optimal PID controller using hybrid differential evolution and particle swarm optimization with an aging leader and challengers, Appl. Soft Comput. 38 (2016) 727–737.
[6] I. Podlubny, Fractional-order systems and $PI^{\lambda}D^{\mu}$–controllers, IEEE Trans. Autom. Control 44 (January (1)) (1999) 208–214.
[7] J.A. Tenreiro Machado, Optimal tuning of fractional controllers using genetic algorithms, Nonlinear Dyn. 62 (1) (2010) 447–452.
[8] J.T. Machado, A.M. Galhano, A.M. Oliveira, J.K. Tar, Optimal approximation of fractional derivatives through discrete-time fractions using genetic algorithms, Commun. Nonlinear Sci. Numer. Simul. 15 (3) (2010) 482–490.
[9] Y. Mousavi, A. Alfi, A memetic algorithm applied to trajectory control by tuning of fractional order proportional-integral-derivative controllers, Appl. Soft Comput. 36 (2015) 599–617.
[10] Y. Bai, H. Zhuang, Z.S. Roth, Fuzzy logic control to suppress noises and coupling effects in a laser tracking system, IEEE Trans. Control Syst. Technol. 13 (January (1)) (2005) 113–121.
[11] I. Baturone, F. Moreno-Velo, S. Sanchez-Solano, A. Ollero, Automatic design of fuzzy controllers for car-like autonomous robots, IEEE Trans. Fuzzy Syst. 12 (August (4)) (2004) 447–465.
[12] P. Bonissone, V. Badami, K. Chiang, P. Khedkar, K. Marcelle, M. Schutten, Industrial applications of fuzzy logic at general electric, Proc. IEEE 83 (March (3)) (1995) 450–465.
[13] Z. Bingul, G. Cook, A. Strauss, K. Rashid, Application of fuzzy logic to spatial thermal control in fusion welding, in: Industry Applications Conference, 1999. Thirty-Fourth IAS Annual Meeting. Conference Record of the 1999 IEEE, vol. 1, 1999, pp. 627–634.
[14] R. Boukezzoula, S. Galichet, L. Foulloy, Observer-based fuzzy adaptive control for a class of nonlinear systems: real-time implementation for a robot wrist, IEEE Trans. Control Syst. Technol. 12 (May (3)) (2004) 340–351.
[15] W.-J. Chang, P.-H. Chen, Intelligent autonomous systems 12 Stabilization for Truck-Trailer Mobile Robot System via Discrete LPV T-S Fuzzy Models, vol. 193, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 209–217.
[16] L.E. Ramos-Velasco, O.A. Domnguez-Ramrez, V. Parra-Vega, Wavenet fuzzy PID controller for nonlinear MIMO systems: experimental validation on a high-end haptic robotic interface, Appl. Soft Comput. 40 (2016) 199–205.
[17] S. Fidanova, P. Pop, An improved hybrid ant-local search algorithm for the partition graph coloring problem, J. Comput. Appl. Math. 293 (2016) 55–61.
[18] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, IEEE Trans. Syst. Man Cybern. SMC-3 (January (1)) (1973) 28–44.
[19] E. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, Proc. Inst. Electr. Eng. 121 (December (12)) (1974) 1585–1588.
[20] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Trans. Syst. Man Cybern. SMC-15 (January (1)) (1985) 116–132.
[21] G. Feng, A survey on analysis and design of model-based fuzzy control systems, IEEE Trans. Fuzzy Syst. 14 (October (5)) (2006) 676–697.
[22] R.E. Precup, L.T. Dioanca, E.M. Petriu, M.B. Rdac, S. Preitl, C.A. Drago, Tensor product-based real-time control of the liquid levels in a three tank system, in: 2010 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, July, 2010, pp. 768–773.
[23] P. Baranyi, P. Korondi, K. Tanaka, Parallel distributed compensation based stabilization of a 3DOF RC helicopter: a tensor product transformation based approach, J. Adv. Comput. Intell. Intell. Inform. 13 (2009) 25–34.
[24] P. Grf, P. Baranyi, P. Korondi, Determination of the stability parameter space of a two dimensional aeroelastic system, a TP model-based approach, in: 2010 IEEE 14th International Conference on Intelligent Engineering Systems, May, 2010, pp. 265–270.
[25] R.E. Precup, C.A. Dragos, S. Preitl, M.B. Radac, E.M. Petriu, Novel tensor product models for automatic transmission system control, IEEE Syst. J. 6 (September (3)) (2012) 488–498.
[26] P. Angelov, R. Yager, Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density, in: 2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS), April, 2011, pp. 62–69.
[27] P. Angelov, I. Škrjanc, S. Blažič, Robust evolving cloud-based controller for a hydraulic plant, in: 2013 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), April, 2013, pp. 1–8.
[28] I. Škrjanc, S. Blažič, P. Angelov, Robust evolving cloud-based PID control adjusted by gradient learning method, in: 2014 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS), June, 2014, pp. 1–8.
[29] B. Costa, I. Škrjanc, S. Blažič, P. Angelov, A practical implementation of self-evolving cloud-based control of a pilot plant, in: 2013 IEEE International Conference on Cybernetics (CYBCONF), June, 2013, pp. 7–12.
[30] S. Blažič, D. Dovžan, I. Škrjanc, Cloud-based identification of an evolving system with supervisory mechanisms, in: 2014 IEEE International Symposium on Intelligent Control (ISIC), October, 2014, pp. 1906–1911.
[31] G. Andonovski, S. Blažič, P. Angelov, I. Škrjanc, Analysis of adaptation law of the robust evolving cloud-based controller, in: 2015 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS), December, 2015, pp. 1–7.
[32] J.G. Ziegler, N.B. Nichols, Optimum settings for automatic controllers, Trans. ASME 64 (1942).
[33] G. Cohen, G. Coon, Theoretical consideration of retarded control, Trans. ASME 75 (1953) 827–834.
[34] K.J. Åström, T. Hägglund, PID Controllers: Theory, Design, and Tuning, 2nd ed., Instrument Society of America, Research Triangle Park, NC, 1995.
[35] H. Kaufman, I. Barkana, K. Sobel, Direct Adaptive Control Algorithms, Springer-Verlag, New York, 1998.
[36] C. Rohrs, L. Valavani, M. Athans, G. Stein, Robustness of continuous-time adaptive control algorithms in the presence of unmodeled dynamics, IEEE Trans. Autom. Control 30 (September (9)) (1985) 881–889.
[37] B. Peterson, K. Narendra, Bounded error adaptive control, IEEE Trans. Autom. Control 27 (December (6)) (1982) 1161–1168.
[38] G. Kreisselmeier, K. Narendra, Stable model reference adaptive control in the presence of bounded disturbances, IEEE Trans. Autom. Control 27 (December (6)) (1982) 1169–1175.
[39] P. Ioannou, P. Kokotovic, Instability analysis and improvement of robustness of adaptive control, Automatica 20 (5) (1984) 583–594.
[40] K. Narendra, A. Annaswamy, A new adaptive law for robust adaptation without persistent excitation, American Control Conference, June (1986) 1067–1072.
[41] P. Ioannou, K.S. Tsakalis, A robust direct adaptive controller, IEEE Trans. Autom. Control 31 (November (11)) (1986) 1033–1043.
[42] I. Škrjanc, D. Matko, Predictive functional control based on fuzzy model for heat-exchanger pilot plant, IEEE Trans. Fuzzy Syst. 8 (December (6)) (2000) 705–712.
[43] G. Andonovski, S. Blažič, P. Angelov, I. Škrjanc, Robust evolving cloud-based controller in normalized data space for heat-exchanger plant, in: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), August, 2015, pp. 1–7.