Robust Evolving Cloud-based Controller (RECCo)

Goran Andonovski
Faculty of Electrical Engineering
University of Ljubljana, Slovenia
goran.andonovski@fe.uni-lj.si

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

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

***Abstract*—This paper presents an autonomous Robust Evolving Cloud-based Controller (RECCo). The control algorithm is a fuzzy type with non-parametric (cloud-based) antecedent part and adaptive PID-R consequent part. The procedure starts with zero clouds (fuzzy rules) and the structure evolves during performing the process control. The PID-R parameters of the first cloud are initialized with zeros and furthermore, they are adapted on-line with a stable adaptation mechanism based on Lyapunov approach. The RECCo controller does not require any mathematical model of the controlled process but just basic information such as input and output range and the estimated value of the dominant time constant. Due to the problem space normalization the design parameters are fixed. The proposed controller with the same initial design parameters was tested on two different simulation examples. The experimental results show the convergence of the adaptive parameters and the effectiveness of the proposed algorithm.**

I. Introduction

The robust evolving cloud-based controller (RECCo) is a type of ANYA fuzzy rule-based (FRB) system [1] with non-parametric antecedents (IF part). This method applies the concept of 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.

Originally, the evolving part of the method relied on local and global data densities [2], [3]. In [4] authors presented an evolving mechanism based just on local density but they introduced an additional parameter. This parameter was fixed in [5] due to the normalization of the problem space.

Since now, there were presented control experiments using RECCo on simulated and real processes to show the effectiveness of the proposed method [5]–[8]. Beside this, one more paper is already accepted for publishing this year where a practical implementation on a real two-tank pilot plant is presented. We need to note that for all this experiments the same design parameters were used.

In this paper we present two simulated non-linear processes with different dynamics. In both examples the same parameters were used.

The following of the paper is organized as follows. In Section II the RECCo control algorithm is described. In Section III the experimental results for both systems are presented. At the end, in Section IV the conclusions are given.

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

II. Robust Evolving Cloud-Based Controller (RECCo)

In this section the RECCo controller will be described. The control algorithm consists of three parts: reference model, evolving law, and adaptation law. The whole control structure is 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 (e.g. input range $[u_{min}, u_{max}]$, output range $[y_{min}, y_{max}]$, time constant τ and sampling time T_s). After the initialization, for every incoming sample, if the certain conditions are satisfied, a new data cloud (fuzzy rule) is added and the controller gains are adapted.

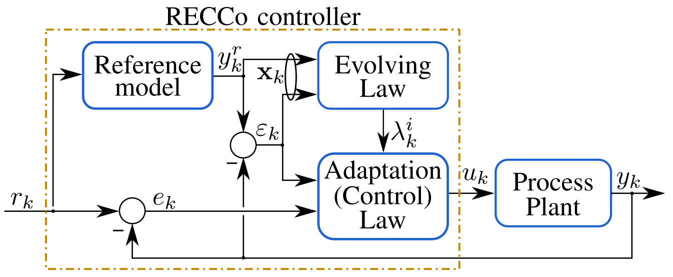

Fig. 1. Control scheme of the RECCo algorithm.

In the following all three parts of the controller will be explained.

A. Reference model

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:

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

where the parameter a_r is the pole of that model. It can be approximated by $(1 - \frac{T_s}{\tau})$, where T_s is the sampling period of the process and τ is the time constant; r_k is the reference signal. The goal of the controller, is to provide efficient performance and to ensure that the tracking error is:

$$\varepsilon_k = y_k^r - y_k \tag{2}$$

as small as possible.

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.

B. Evolving law

The evolving law tackles the fuzzy structure of the controller. The RECCo algorithm is based on the ANYA FRB system proposed in [1] and has the following form:

$$\mathcal{R}^i : \quad \text{IF} \quad (\mathbf{x} \sim X^i) \quad \text{THEN} \quad (u^i) \tag{3}$$

where the number of rules $\mathcal{R}^i$ is equal to the number of clouds in the data space $i = 1, \dots, c$. The antecedent part, where the evolving low has the main role, is defined with the operator $\sim$ which could be linguistically expressed as '*is associated with*'. That means that the current data $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ is related to the i-th cloud $X^i \in \mathbb{R}^n$. The consequent part is defined by c local control actions u^i.

The data vector is defined in a 2-dimensional space as follows:

$$\mathbf{x}_k = \left[\begin{array}{cc} \frac{\varepsilon_k}{\Delta\varepsilon}, & \frac{y_k^r - y_{min}}{\Delta y} \end{array} \right]^T \tag{4}$$

where $\Delta y = y_{max} - y_{min}$ and $\Delta\varepsilon = \frac{\Delta y}{2}$. In this case the data point is normalized and due to this the evolving parameter can be fixed. Please refer to [5], [6] for more details about the problem space normalization.

The data clouds X^i have 4 properties: mean value μ^i, mean-square length σ^i, number of data points M^i and time stamp when it is added k_{add}^i. The degree of association between the data sample $\mathbf{x}$ and corresponding cloud X^i is measured by the normalized relative density as follows:

$$\lambda_k^i = \frac{\gamma_k^i}{\sum_{j=1}^{c} \gamma_k^j} \quad i = 1, \dots, c \tag{5}$$

where γ_k^i is the local density of the i-th cloud for the current data $\mathbf{x}_k$ and is defined in a recursive form as follows:

$$\gamma_k^i = \frac{1}{1 + \|\mathbf{x}_k - \mu_k^i\|^2 + \sigma_k^i - \|\mu_k^i\|^2} \tag{6}$$

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 in the i-th cloud. Both of them can be recursively calculated using following equations for mean value and mean-square length, respectively:

$$\mu_k^i = \frac{M^i - 1}{M^i} \mu_{k-1}^i + \frac{1}{M^i} \mathbf{x}_k \tag{7}$$

$$\sigma_k^i = \frac{M^i - 1}{M^i} \sigma_{k-1}^i + \frac{1}{M^i} \|\mathbf{x}_k\|^2 \tag{8}$$

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 only a mechanism for adding new clouds (rules). Beside this, another evolving mechanisms such as merging, splitting and removing clouds can be also implemented [9]. We decide to use just adding mechanism due to simplicity of the implementation and because it is sufficient for control the plant presented in next section.

The adding mechanism relies on the local density γ_k^i of the current data sample with the existing clouds. 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 equations (7) and (8). 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. Due to the normalization of the problem space 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, 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.

C. Adaptation law

For the consequent part of the RECCo controller the PID-R type control is used [4] and to each cloud (fuzzy rule) a local PID-R controller is assigned with the following form:

$$u_k^i = P_k^i \varepsilon_k + I_k^i \Sigma_k^\varepsilon + D_k^i \Delta_k^\varepsilon + R_k^i, \quad i = 1, \dots, c \tag{9}$$

where P_k^i, I_k^i, D_k^i are controller gains while R_k^i is compensation of the operating point. Σ_k^ε and Δ_k^ε in (9) are discrete-time integral and derivative of the tracking error, respectively, and can be calculated as follows:

$$\Sigma_k^\varepsilon = \begin{cases} \Sigma_{k-1}^\varepsilon + \varepsilon_k, & u_{min} < u(k) < u_{max} \\ \Sigma_{k-1}^\varepsilon, & u(k) = u_{min} \textbf{ or } u(k) = u_{max} \end{cases} \tag{10}$$

$$\Delta_k^\varepsilon = \varepsilon_k - \varepsilon_{k-1} \tag{11}$$

The discrete-time integral is additionally protected with anti-windup mechanism for protecting the integral explosion.

The vector of the PID-R parameters is denoted as $\theta_k^i = [P_k^i, I_k^i, D_k^i, R_k^i]^T$. The vector of the first cloud is initialized with zeros $\theta_0^1 = [0, 0, 0, 0]^T$, while newly added clouds is initialized with weighted mean value of the parameters of all previous clouds as follows:

$$\theta_{k_{add}^c}^c = \sum_{j=1}^{c-1} \lambda_k^j \theta_k^j \tag{12}$$

where c is the index of the newly added cloud and k_{add}^c is equal to current time stamp k.

The adaptation of the PID-R parameters is made in an online manner and only the parameters of the active cloud are updated while others are kept constant:

$$\theta_k^i = \theta_{k-1}^i + \Delta\theta_k^i \tag{13}$$

The adaptation law for each parameter is defined as follows:

$$\begin{aligned}
\Delta P_k^i &= \alpha_P\, G_{sign} \lambda_k^i \frac{|e_k \varepsilon_k|}{1+r_k^2} \\
\Delta I_k^i &= \alpha_I\, G_{sign} \lambda_k^i \frac{|e_k \Delta_k^\varepsilon|}{1+r_k^2} \\
\Delta D_k^i &= \alpha_D\, G_{sign} \lambda_k^i \frac{|e_k \Delta_k^\varepsilon|}{1+r_k^2} \\
\Delta R_k^i &= \alpha_R\, G_{sign} \lambda_k^i \frac{\varepsilon_k}{1+r_k^2}
\end{aligned} \tag{14}$$

where $\alpha_P, \alpha_I, \alpha_D, \alpha_R$ are the adaptation gains of the controller parameters, $G_{sign} = \pm 1$ is the known sign of the process gain, $e_k = r_k - y_k$ is the control error. The default value of the adaptation gains 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 rescaled as follows:

$$\alpha_{new} = \frac{u_{max} - u_{min}}{20} \cdot 0.1 \tag{15}$$

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 (14) 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. Please refer to [10] for more details about the absolute values in (14).

The adaptive law in (14) uses the product of the tracking and the process error as a cost function with normalization (part $1 + r_k^2$ in (14)). Please refer to Chapter 4.3 in [11] for more details about adaptive laws with normalization and how the stability (Lyapunov approach) of such adaptation is explained.

Finally, after the adaptation of the parameters is performed, for the defuzzification the weighted average is used (but not limited to this form) and therefore, the control variable becomes:

$$u_k = \sum_{i=1}^{c} \lambda_k^i u^i = \frac{\sum\limits_{i=1}^{c} \gamma_k^i u^i}{\sum\limits_{i=1}^{c} \gamma_k^i} \tag{16}$$

where u^i denotes the contribution of the i-th local controller.

D. The instability protection mechanism

This subsection is devoted to the modifications of the adaptation law (13)–(14) 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. In this paper we will use the following techniques:

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 [12]:

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

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) Parameter projection: Parameter projection mechanism is used to guarantee that the estimation of the parameters will stay within finite known region [13]. 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 (13) is generalized as follows:

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

In our case we chose $\underline{\theta} = 0$ and $\overline{\theta} = \infty$ 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 $\underline{\theta} = -\infty$.

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 [14].

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. The value of leakage used in this paper is $\sigma_L = 10^{-6}$.

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 this 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:

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

Finally, we can summarize the whole procedure of RECCO controller presented above in the pseudo Algorithm 1.

III. Experimental results

In this section we present the effectiveness of the RECCo controller through two examples. The first example is an artificial first order process with quadratic static nonlinearity. The second example is a hydraulic pilot plant of two tanks linked by a valve. Please fine more details in [4] about the both processes.

Algorithm 1 Pseudo code of the RECCo PID control algorithm

```
1: Initialize (Process parameters): $\tau$, $T_s$, $u_{min}$, $u_{max}$, $y_{min}$, $y_{max}$.
2: Initialize (Evolving parameters): $\gamma_{max} = 0.93$, $c = 0$, $c_{max} = 20$, $n_{add} = 20$.
3: Initialize (Adaptation parameters): $\alpha_R$, $\alpha_P$, $\alpha_I$, $\alpha_D$, $d_{dead}$, $\underline{\theta}$, $\overline{\theta}$, $\sigma_L$.
4: repeat
5:     Measurement: $y_k$.
6:     Define and compute: $y_k^r$ using (1)   ▷ Reference model
7:     Compute: $e_k$, $\varepsilon_k$, $\Sigma_k^\varepsilon$, $\Delta_k^\varepsilon$.
8:     Compute: $\mathbf{x}_k$ using (4).
9:     if $c = 0$ then   ▷ Start of the evolving law
10:        Increment: $c$,
11:        Store: $k_{add}^c$,
12:        Initialize: $\mu_0^1$, $\sigma_0^1$, $\theta_0^1$.
13:    else
14:        Calculate: $\gamma_k^i$, $\lambda_k^i$, for $i = 1, \ldots, c$
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: $\mu_0^c$, $\sigma_0^c$, $\theta_0^c$.
19:        else
20:            Associate sample $\mathbf{x}_k$ with cloud ($max_i \gamma_k^i$)
21:            Update $\mu_k^i$, $\sigma_k^i$ 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 using (14).
25:    Computation of the control law using (16).
26:    Check the protection mechanisms (17), (18), (19), (20).
27: until End of data stream.
```

A. First example

$$
\begin{aligned}
y_k^p &= a_p y_{k-1}^p + b_p u_k \\
y_k &= (y_k^p)^2 + n_k
\end{aligned}
\tag{21}
$$

where y_k^p 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 input and output range of the process are $[u_{min}, u_{max}] = [0, 10]$ and $[y_{min}, y_{max}] = [0, 10]$, respectively. The sampling time is $T_s = 1\,\mathrm{s}$ and the desired time constant is $\tau = 40\,\mathrm{s}$. The results of the first experiment are presented in Fig. 2. The reference, model reference, controlled and control signals are shown (for the starting and the finishing phase of the experiment). The tracking error is shown for the whole experiment. On the right in Fig. 2 the created clouds are shown. In this experiment only tree clouds were created. The four plots on the bottom in Fig. 2 show the adaptation of the controller parameters P_k^i, I_k^i, D_k^i and R_k^i. It could be considered that the adaptation law converge through time.

B. Second example

$$
\begin{aligned}
\xi_{k-1} &= y_{k-1}^p - y_{k-1} \\
y_k^p &= y_{k-1}^p + T_s a_p \left(u_k - k_{v1} * \sqrt{|\xi_{k-1}|} sign(\xi_{k-1}) \right) \\
\xi_k &= y_k^p - y_{k-1} \\
y_k &= y_{k-1} + T_s b_p \left(k_{v1} \sqrt{|\xi_k|} sign(\xi_k) - k_{v2} \sqrt{y_{k-1}} \right)
\end{aligned}
\tag{22}
$$

where y_k^p stands for intermediate variable, $a_p = 0.02$, $b_p = 0.02$, $k_{v1} = 3$, $k_{v2} = 1$ are the process parameters and n_k stands for output noise with characteristics $\mathcal{N}(0, 0.005)$. The input and output range of the process are $[u_{min}, u_{max}] = [0, 5]$ and $[y_{min}, y_{max}] = [0, 5]$, respectively. The sampling time is $T_s = 1\,\mathrm{s}$ and the desired time constant is $\tau = 40\,\mathrm{s}$. The results of the first experiment are presented in Fig. 3. The reference, model reference, controlled and control signals are shown (for the starting and the finishing phase of the experiment). The tracking error is shown for the whole experiment. On the right in Fig. 3 the created clouds are shown. In this experiment only four clouds were created. The four plots on the bottom in Fig. 3 show the adaptation of the controller parameters P_k^i, I_k^i, D_k^i and R_k^i. It could be considered that the adaptation law converge through time.

IV. Conclusion

In this paper we present the Robust Evolving Cloud-based Controller (RECCo) for two different simulation processes. Despite the deviation between them the RECCo algorithm efficiently controlled both of them using the same design parameters. Only the basic information of the controlled process are required, such as input and output range and time constant. The controller starts with empty structure and is initialized with the first data point received. After the initialization phase the structure evolves when certain criteria are fulfilled. The controller's parameters are adapted in online manner using stable steepest decent mechanism based on Lyapunov approach.

References

[1] P. Angelov and R. Yager, "Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density," in *Symposium Series on Computational Intelligence (IEEE SSCI 2011) - IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS 2011)*, 2011, pp. 62–69.

[2] P. Angelov, I. Škrjanc, and S. Blažič, "Robust Evolving Cloud-Based Controller for a Hydraulic Plant," in *2013 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 2013, pp. 1–8.

[3] B. S. J. Costa, I. Škrjanc, S. Blažič, and P. Angelov, "A practical implementation of self-evolving cloud-based control of a pilot plant," in *IEEE International Conference on Cybernetics, CYBCONF*, 2013, pp. 7–12.

[4] I. Škrjanc, S. Blažič, and P. Angelov, "Robust Evolving Cloud-based PID Control Adjusted by Gradient Learning Method," *2014 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pp. 1–8, 2014.

[5] G. Andonovski, S. Blažič, P. Angelov, and 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)*, Istanbul, Turkey, 2015, pp. 1–7.

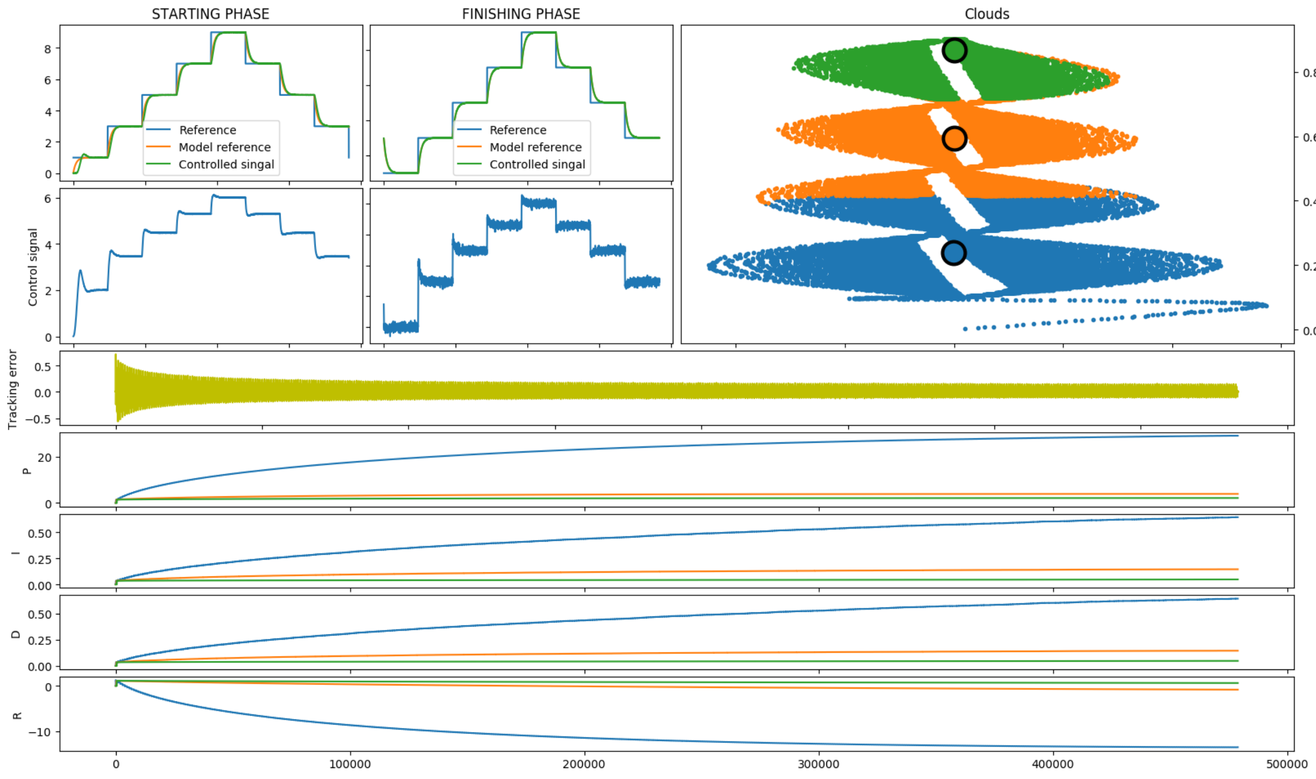

Fig. 2. Simulation results of the first example.

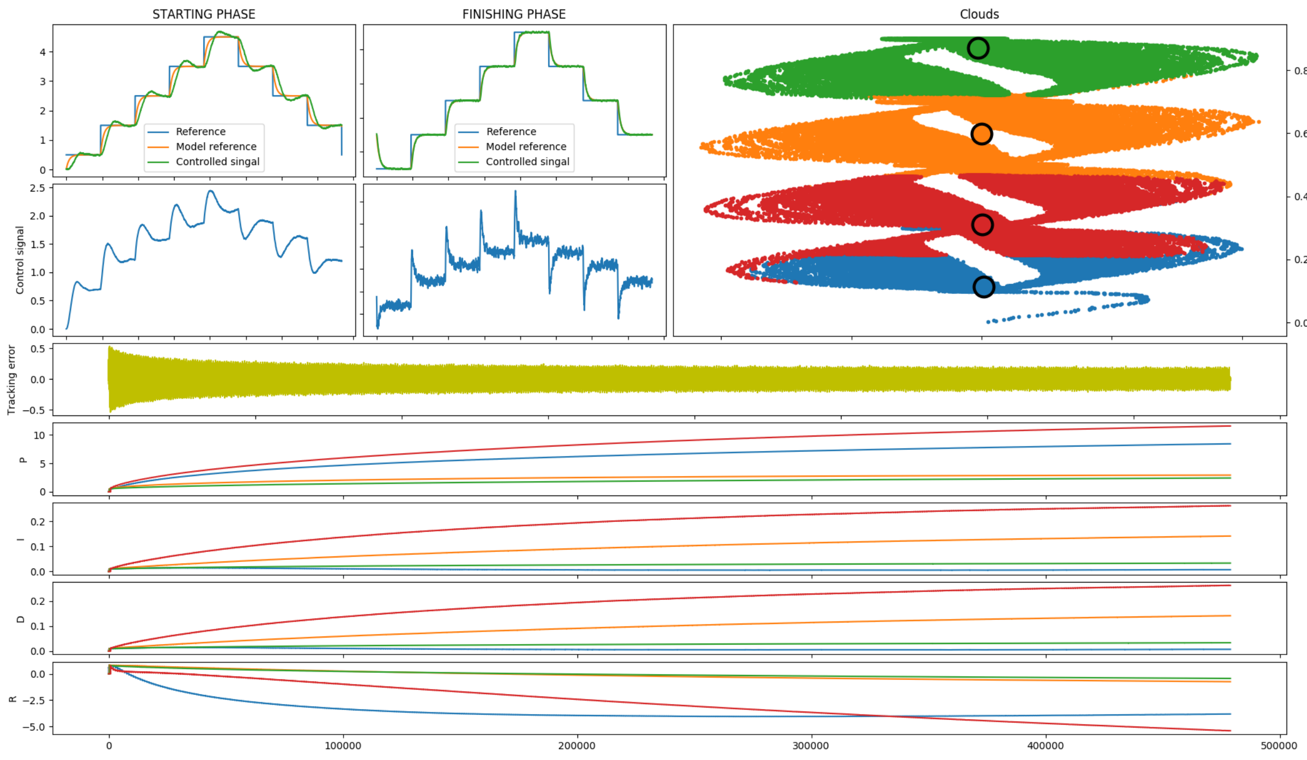

Fig. 3. Simulation results of the second example.

[6] G. Andonovski, P. Angelov, S. Blažič, and I. Škrjanc, "A practical implementation of Robust Evolving Cloud-based Controller with normalized data space for heat-exchanger plant," *Applied Soft Computing*, vol. 48, pp. 29–38, 2016.

[7] G. Andonovski, A. Bayas, and S. Doris, "Robust Evolving Cloud-based Control for the Distributed Solar Collector Field," in *2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2016, pp. 1570–1577.

[8] G. Andonovski, E. Lughofer, and I. Škrjanc, "A Comparison of RECCo and FCPFC Controller on Nonlinear Chemical Reactor," *Modelling, Identification and Control*, pp. 214–221, 2017.

[9] G. Andonovski, G. Mušič, S. Blažič, and I. Škrjanc, "On-line Evolving Cloud-based Model Identification for Production Control," *IFAC-PapersOnLine*, vol. 49, no. 5, pp. 79–84, 2016.

[10] G. Andonovski, S. Blažič, P. Angelov, and 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)*, 2015, pp. 1–7.

[11] P. Ioannou and J. Sun, *Robust Adaptive Control*. PTR Prentice-Hall, 1996.

[12] B. B. Peterson and K. S. Narendra, "Bounded Error Adaptive Control," *IEEE Transactions on Automatic Control*, vol. 27, no. 6, pp. 1161–1168, 1982.

[13] G. Kreisselmeier and K. Narendra, "Stable model reference adaptive control in the presence of bounded disturbances," *IEEE Transactions on Automatic Control*, vol. 27, no. 6, pp. 1169–1175, 1982.

[14] P. Ioannou and P. Kokotovic, "Instability analysis and improvement of robustness of adaptive control," *Automatica*, vol. 20, no. 5, pp. 583–594, 1984.