Hindawi
Complexity
Volume 2017, Article ID 9241254, 8 pages
https://doi.org/10.1155/2017/9241254

*Research Article*

# A Gain-Scheduling PI Control Based on Neural Networks

**Stefania Tronci and Roberto Baratti**

*Dipartimento di Ingegneria Meccanica, Chimica e dei Materiali, Università degli Studi di Cagliari, Via Marengo 2, 09123 Cagliari, Italy*

Correspondence should be addressed to Stefania Tronci; stefania.tronci@dimcm.unica.it

Received 13 July 2017; Revised 19 September 2017; Accepted 24 September 2017; Published 19 October 2017

Academic Editor: Jing Na



This paper presents a gain-scheduling design technique that relies upon neural models to approximate plant behaviour. The controller design is based on generic model control (GMC) formalisms and linearization of the neural model of the process. As a result, a PI controller action is obtained, where the gain depends on the state of the system and is adapted instantaneously on-line. The algorithm is tested on a nonisothermal continuous stirred tank reactor (CSTR), considering both single-input single-output (SISO) and multi-input multi-output (MIMO) control problems. Simulation results show that the proposed controller provides satisfactory performance during set-point changes and disturbance rejection.

## 1. Introduction

Most industrial plants are nonlinear in nature, but process control often relies on traditional linear PID algorithm, because of its simplicity and well recognition by the industry. This solution can be justified by the fact that nonlinear processes can be approximated with a linear model as they approach steady state; therefore, in case of close regulatory control, the use of a linear control is adequate [1]. Nevertheless, for those nonlinear processes whose nonlinearities are strong and large changes of the operating conditions occur during the operation, linear controller designs are inadequate and more effective alternatives should be considered [2]. The availability of powerful computer tools opened the way to implement advanced process control strategies, where system nonlinearities can be taken into account.

Model-based feedback control can be a valid alternative to the use of the linear model-free PI algorithm, and the use of adaptive parameters along the process motions can represent a possible solution for controlling nonlinear time-variant processes [3]. Adaptive systems are also used to compensate input delay, as proposed in Na et al. [4] where an adaptive NN observer was designed for nonlinear systems. The generic model control (GMC) of Lee and Sullivan [5] is probably one of the simplest nonlinear control techniques to install and maintain among nonlinear model-based controllers. In the chemical engineering field the application of GMC strategy was investigated for control reactors [6, 7], batch cooling crystallizers [8], batch and semibatch polymerization reactors [9, 10], and, recently, multistage flash desalinations [11].

In this work, the GMC algorithm was used in conjunction with a dynamic neural network model, which describes the nonlinear relationship between controlled outputs and manipulated inputs [12, 13]. The main advantage of the proposed algorithm is the obtainment of a PID-like controller structure, where the nonlinear dependence of the process gain on the operating condition is achieved by a gain-scheduling control scheme. The proposed algorithm is simple to implement and it can be obtained without the need of a detailed knowledge of the plant; therefore it is suited for industrial applications where standard solutions are generally preferred. The performance of the proposed technique for nonlinear process control was tested for two case studies, referring to a SISO [14] and a MIMO [15] control problem for a nonisothermal continuously stirred tank reactor (CSTR). These two cases, which are well known benchmarks for testing control methodology, were selected because the strong nonlinearities, due to the Arrhenius dependence of the kinetic rate on temperature, lead to a very rapid response of the process variables in regions of high conversion and a very

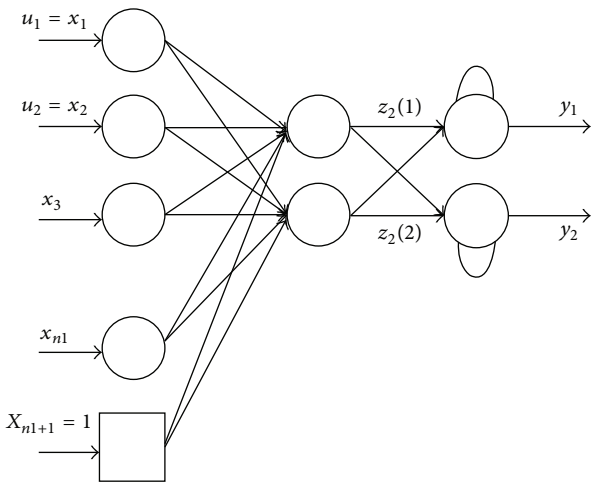


Figure 1: Neural network structure.

mild response in regions of low conversion. Therefore, the need for adaptive control strategies is more evident.

## 2. Neural Network Model

There are several possibilities of building a dynamic neural network that may be classified in two main classes: time-lagged feedforward networks, TLFN, or recurrent neural networks, RNN (cf. [16]). While in the TLFN the dynamics are generally accounted for in the input layer as a linear combination of inputs (present and past values of the inputs) followed by a static nonlinear mapper between inputs and outputs, in the RNN the memory mechanism is brought inside the nonlinear mapping; that is, recurrent connections are applied among some or all layers (cf. Jordan or Elman networks). Both the architectures are able to describe the dynamic behaviour of a process, while training procedure and stability become an issue moving from the TLFN to the RNN (cf. [16]).

In this work, a multi-feed-forward neural network with recurrent neurons in the output layer is used to describe the process dynamics. The net topology is shown in Figure 1 and can be considered a special case of the TLFN where the dynamic is moved from the input to the output layer and, in this way, the advantage of the TLFN over the RNN is maintained.

The equations that describe the neuron output evolution are reported in the following:

$$\tau \frac{dy_k}{dt} + y_k = \sum_{j=1}^{n_2} w_2(j,k)\, z_2(j), \tag{1a}$$

$$z_2(j) = f\left(\sum_{i=1}^{n_1+1} w_1(i,j)\, z_1(i)\right), \tag{1b}$$

where the activation function $f(\cdot)$ is chosen as

$$f(x) = \frac{1}{1+\exp(-x)}, \quad x \in \mathscr{R}. \tag{2}$$

The weight $w_2(j,k)$ is the interconnection between the $k$th output and the $j$th hidden neuron; $w_1(i,j)$ is the connection between the $i$th input and the $j$th hidden neuron; $\mathbf{z_2}$ is the output vector from the hidden layer; and $\mathbf{z_1}$ is the input vector. It is worth noting that the nonlinear plant characteristic modelled by (1a)-(1b) is stored in the weights between the input and output layers, and this represents the long-term capability prediction, while the locally recurrent neurons in the output layer may be thought of as a representational layer for time (short-term) of information (cf. [16]).

In the present work, this neural model will be applied to develop a gain-scheduling control scheme. In order to obtain the simplest controller structure, a number of hidden neurons equal to the number of manipulated variables are selected. As results reported in the following sections will demonstrate, this choice does not affect the neural model performance. Under this assumption, for a SISO system the neural network expression becomes

$$\tau \frac{dy}{dt} + y = w_2(1,1)\, z_2(1) = w_2(1,1)\, f(u, \mathbf{x}), \tag{3}$$

where $y$ is the predicted output and $u$ is the manipulated variable. The variable $z_2(1)$ is the only output of the hidden neuron, which nonlinearly depends on the unique manipulated variable, $u$, since sigmoidal activation function is used for neurons in the hidden layer. The vector $\mathbf{x}$ represents the other net inputs necessary to describe the plant properly. For the sake of brevity, the case of the neural network for a MIMO system is not reported, since it can be straightforwardly derived from (3).

## 3. Neural Based Gain-Scheduling Control

Starting from the nonlinear neural model, a GMC [5] approach is used to synthesize the gain-scheduling control. In order to define the controller action, the desired output behaviour has to be specified in the form of a trajectory. Then the process model is used directly to synthesize the controller required to cause the process output to follow this trajectory. A good choice [5] for the reference trajectory $y_r$ is reported in

$$\frac{dy_r}{dt} = K_1(y_d - y) + K_2 \int (y_d - y)\, dt, \tag{4}$$

where, for a given desired output $y_d$, a suitable selection of parameters $K_1$ and $K_2$ can be made to achieve a variety of responses in $y(t)$. Substituting the reference trajectory in (3), the action of the manipulated variable can be implicitly derived from the model. Of course, the control action in this case is not linear, and it is difficult to implement. Referring to Ogunnaike and Ray [17], a GMC control structure can lead to a PI-like controller if a first-order system is driven to follow a first-order reference trajectory, that is, setting $K_2 = 0$ in (4).

Letting $y = y_r$, the process model (3) can be used directly to obtain the controller required to cause the process to follow trajectory (4) and solving for $u(t)$. It should be pointed out that using model (3) no explicit solution is available for the manipulated input and the controller equation has

to be solved numerically. The controller algorithm can be simplified by linearizing the right-hand side of (3); hence the model for a SISO system becomes

$$\tau \frac{dy}{dt} + y = w_2(1,1) \frac{\partial z_2}{\partial u} u(t). \tag{5}$$

In the following, the coefficient of the manipulated input will be indicated with $K$ (6). The gain $K$ depends on the state of the system, through the derivative of $z_2$ with respect to the input of the neural model:

$$K = w_2(1,1) \frac{\partial z_2}{\partial u}. \tag{6}$$

The nonlinearity of the system is taken into account by calculating at every sampling time the derivative of $z_2$ with respect to the inputs from the neural model (3).

The PI controller action is derived at this point by setting $K_2 = 0$ in the reference trajectory (4), leading to

$$\frac{dy_r}{dt} = K_1 (y_d - y), \tag{7}$$

or

$$\frac{dy_r}{dt} = K_1 e(t), \tag{8}$$

where $e(t)$ represents the usual feedback error term. Integrating (8), the following expression for the desired trajectory is obtained:

$$y_r = K_1 \int e(t)\, dt. \tag{9}$$

The control action required to cause the process to follow the trajectory described by (8) and (9) may be obtained by substituting the desired trajectory expressions in the linearized model (5), leading to

$$\tau K_1 e(t) + K_1 \int e(t) = K u(t) \tag{10}$$

and then solving for $u(t)$:

$$u(t) = \frac{\tau K_1}{K} \left[ e(t) + \frac{1}{\tau} \int_0^t e(\theta)\, d\theta \right]. \tag{11}$$

In this way, a PI controller law is obtained, where $e$ represents the actual error, $\tau$ is equal to the time constant of the output neuron, $K_1$ is a tuning parameter and is set as the inverse of the desired time of response, and $K$ is the static gain of the linear first-order system that we have obtained by linearizing the neural model of the system, as defined by (5). The gain is a function of the system status because it depends on the derivative of $z_2$ with respect to the input. The presence of the integral term will compensate for the error of the approximated neural model.

The advantage of such an approach is that the nonlinear behaviour of the process is considered because of the time-varying nature of the controller gain. As a result, since parameters are simply adjusted on-line for all process conditions, a standard PI controller structure is maintained. The proposed methodology has the scheme of a gain-scheduling control approach, as the block diagram in Figure 2 shows. It is important to note that in this case the use of a neural model removes the need for detailed process knowledge to define operating bands and for open loop tests to locally calibrate the controller gain within each band, which is the typical drawback of the gain-scheduling controller scheme.

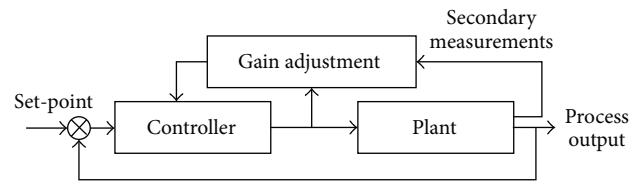


Figure 2: Neural scheduled adaptive control scheme.

## 4. Results and Discussion

As mentioned in Introduction, two continuously stirred tank reactor systems are considered to show the performance of the proposed control algorithm. The two proposed case studies are well known benchmarks for advanced control methodology testing [14, 15, 18].

*4.1. Case 1.* The first case study is the CSTR proposed by Lightbody and Irwin [18] and Ge et al. [14]. The system consists of a constant volume reactor cooled by a single coolant stream, and the objective is to control the effluent concentration, $y_1$, by manipulating the coolant flow rate, $u_1$. The model equations are reported as follows:

$$\frac{dC_a}{dt} = \frac{q}{V}(C_{a0} - C_a) - a_0 C_a e^{-E/RT_a}, \tag{12a}$$

$$\frac{dT_a}{dt} = \frac{q}{V}(T_f - T_a) + a_1 C_a e^{-E/RT_a} + a_3 q_c \left(1 - e^{-a_2/q_c}\right)(T_{cf} - T_a), \tag{12b}$$

where $y_1 = C_a$ and $u_1 = q_c$. The model parameters and nominal values used for this work are reported in Table 1.

The selected neural network model that describes the process is composed of two neurons in the input layer, one neuron in the hidden one (because one is the number of the manipulated variables), and one neuron in the output layer. The system has one natural input, which is the coolant flow rate, $u_1$. In order to provide more information on the state of the reactor, another input, $x_2$, is selected as energy balance around the cooling system. The net is trained using data obtained simulating the reactor for 300 minutes (1500 points sampled every 0.2 minutes), by exciting the plant through step variations of the manipulated variable. The input was modified every 5 minutes, randomly varying the amplitude of the step in the range 94.7–108.0 l/min. The Levenberg-Marquardt algorithm is used to estimate the network's weights, with the sum of squared error as objective function and performing the training for one hundred initial

Table 1: Summary of parameters and variables in the CSTR model for Case 1.

| Symbol | Nominal Value | Description |
|---|---|---|
| $q$ | 100 ml/min | Process flow-rate |
| $C_{a0}$ | 1 mol/l | Concentration of component A |
| $T_f$ | 350 K | Feed temperature |
| $T_{cf}$ | 350 K | Inlet coolant temperature |
| $q_c$ | 100 l/min | Coolant flow-rate |
| $V$ | 100 ml | Volume of tank |
| $h_a$ | $7 \cdot 10^5$ J/(min K) | Heat transfer coefficient |
| $a_0$ | $7.2 \cdot 10^{10}$ min$^{-1}$ | Pre exponential factor |
| $E/R$ | $1 \cdot 10^{10}$ K | Activation energy |
| $-\Delta H$ | $2 \cdot 10^4$ cal/mol | Heat of reaction |
| $\rho_1, \rho_c$ | $1 \cdot 10^3$ g/l | Liquid densities |
| $C_p, C_{pc}$ | 1 cal/(g K) | Heat capacities |
| $a_1 = \dfrac{(-\Delta H)\, a_0}{\rho_1 C_p} = 1.44 \cdot 10^{13}$ | $a_2 = \dfrac{h_a}{\rho_c C_{pc}} = 698.7$ | $a_3 = \dfrac{\rho_c C_{pc}}{\rho_1 C_p V} = 0.01$ |

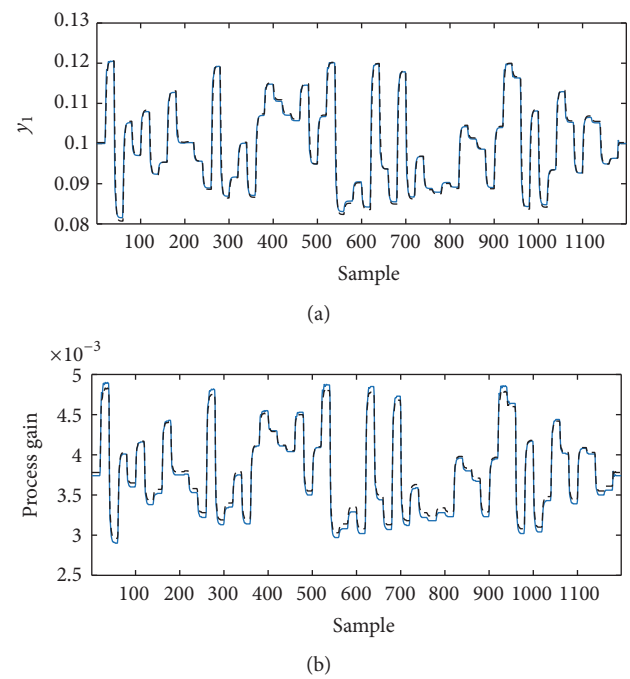


Figure 3: Performance of the neural model to reconstruct concentration and Jacobian for CSTR in Case 1. (a) Concentration predicted by the neural model (black dashed line) and CSTR (blue line). (b) Process gain calculated with the neural model (black dashed line) and actual gain of the process (blue line).

values, randomly generated. The selected weights are the ones leading to the lower error calculated on validation set, which is the 30% of the total amount of data recorded.

The capability of the network to reconstruct the system dynamics is shown in Figure 3, where concentration and Jacobian values estimated by the neural model are compared to those calculated by integrating equations ((12a)-(12b)) of the true plant. In this case, the output was obtained by randomly changing the manipulated variable and corrupting the measured inputs to the network by noise, in particular introducing an error of ±1°C in temperature measurements and ±2 l/min in the coolant flow rate measurement. Results indicate that the neural model reliably represents the system behaviour, at least for the considered process conditions; in fact the line representing the estimated variables completely masks the true values. When the plant model is not available, proof of the ability of the network to reconstruct system gain can be accomplished experimentally by performing an appropriate set of step tests.

The obtained dynamic neural model was then applied to construct the gain-scheduling PI controller described in Section 3. As a result, the only parameter to be tuned is the inverse of the desired time response, $K_1$, because the controller gain and the integral time are derived from the model (the latter is not adjusted in time). In order to guarantee the robustness of the control system, the inverse of $K_1$ is set 2.5 times smaller than the maximum characteristic time of the process, applying the recommended choice for model-based controller that suggests to have the desired closed-loop time constant for a first-order process greater than $0.2\,\tau$ [17]. The good fit shown in Figure 3 between the Jacobian calculated by the neural model and the true one guarantees that the gain will be properly adapted on-line.

The adaptive controller technique was tested in terms of set-point tracking, performing the test and comparing the results to a conventional PI controller, as reported in Ge et al. [14]. The set-point tracking results are shown in Figure 4, where the output variable, $y_1$, and the loads on the manipulated variables, $u_1$, are reported for the adaptive controller and the conventional PI. The results show that the adaptive controller exhibits good set-point tracking capabilities (short response time), without requiring excessive loads on the manipulated variables. Indeed, the curves representing the programmed set-point changes are almost completely masked by the ones representing the dynamic behaviour of the CSTR under the control of the gain scheduled PI. The presence of a low overshoot is the compromise for a short time response. On the other hand, the conventional PI

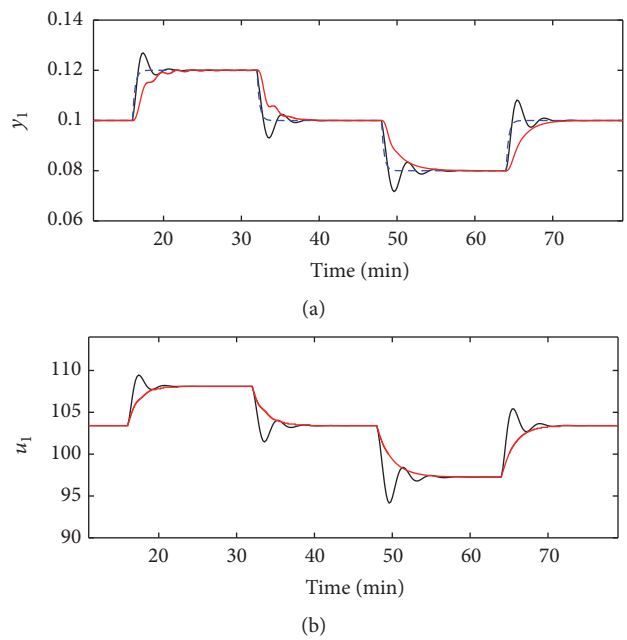


Figure 4: Performance of the gain-scheduling PI in Case 1 and a conventional PI with $K_c = 44.0$ and $\tau_I = 0.25$. (a) Set-point (dashed blue line), gain-scheduling controller (black line), and conventional PI (red line). (b) Manipulated variable for gain-scheduling controller (black line) and conventional PI (red line).

shows a sluggish response with respect to the gain-scheduling controller.

*4.2. Case 2.* As a second case study, a CSTR in which an exothermic first-order reaction takes place is considered. This benchmark was proposed by Scott and Ray [15] in order to demonstrate the inadequacy of a standard PI controller in such nonlinear systems and is described in a dimensionless form by the following differential equations:

$$\frac{dx_1}{dt} = \frac{1}{\tau}\left(-(m_2+1)(x_1-d_2) + Da(1-x_1)e^{\gamma x_2/(x_2+1)}\right), \tag{13a}$$

$$\frac{dx_2}{dt} = \frac{1}{\tau}\left((m_2+1)(d_1-x_2) + \frac{BDa}{\gamma}(1-x_1)e^{\gamma x_2/(x_2+1)} - \beta(x_2-m_1)\right). \tag{13b}$$

According to Scott and Ray [15], the meaning of the constants and variables that appear in (13a) and (13b), along with the normalized variable used for training the network and representing the results of the control system, are summarized in Table 2.

The control objective is to maintain both bulk temperature and concentration at set-point values. The manipulated variables are the coolant temperature, $m_1$, and the input feed flow rate, $m_2$, while the control variables are the reactant mole fraction, $x_1$, and the bulk temperature, $x_2$.

The dynamic neural model of the CSTR consists of a net with four neurons in the input layer, two neurons in the hidden layer, and two neurons in the output layer. Four inputs were selected with the aim of giving the net the most representative information about the process status. Indeed, two terms which represent, respectively, the heat flux exchange through the cooling system and a measure of the heat flux carried out with the convective outlet stream were fed as input to the network, along with the two manipulated variables (cf. [13]).

The dynamic neural network was trained using 1000 data points generated through the CSTR dynamic simulator (sampling time equal to 0.1 min), randomly changing the dimensionless manipulated variable. The Levenberg-Marquardt algorithm was used to calculate the weights using the sum of the squared error as objective function and selecting the best model evaluating the predictions on the validation test. Also in this situation, it was verified that the Jacobian of the neural model fitted the exact one, which is a guarantee of the robustness of the control system. Results are not reported for the sake of brevity.

As in the previous case, the only controller parameter to be tuned is the inverse of the desired time response, $K_1$, because the controller gain and the integral time are derived from the model. In this case, the inverse of $K_1$ is set 2.5 times smaller than the characteristic time of the process for both the controllers [17].

The gain-scheduling PI controller was tested in terms of set-point tracking and disturbance rejection capabilities, performing the same test as in Scott and Ray [15]. The set-point changes and disturbances imposed on the system to test the performance of the proposed methodology are summarized in Table 3.

For the sake of brevity, only the performance of the gain-scheduling control structure at the point of maximum gain is reported in graphical form, while all the other results are reported in Table 4 using the integral squared error (ISE) as performance index:

$$\mathrm{ISE} = \int_0^\infty \left(y_{\mathrm{sp},1} - y_1\right)^2 dt + \int_0^\infty \left(y_{\mathrm{sp},2} - y_2\right)^2 dt. \tag{14}$$

The set-point tracking results are shown in Figure 5(b) for the first variable, $y_1$, and in Figure 5(c) for the second variable, $y_2$, while the loads on the manipulated variables are reported in Figure 5(a). The results show that the controller exhibits good set-point tracking capability (short response time) without requiring excessive loads on the manipulated variables. Indeed, the curves representing the programmed set-point changes (dashed curves) are completely masked by the ones representing the dynamic behaviour of the CSTR under the control of the gain scheduled PI controller. The performance of the controller with respect to disturbance rejection is shown in Figures 5(d)–5(f). Also in this situation the gain-scheduling controller performs well and exhibits good capability to quickly compensate for the upsets entering the CSTR.

As in the previous case, noisy measured inputs fed to the neural network did not affect the performance of the

Table 2: Summary of parameters and variables in the CSTR model for Case 2.

| Symbol | Value or Range | Description |
|---|---|---|
| $\tau = \dfrac{V}{F_0}$ | 1.0 | Nominal space time of reactor |
| $Da = \dfrac{k_0 e^{-\gamma} V}{F_0}$ | 0.11 | Damkohler number |
| $\gamma = \dfrac{E}{\left(RT_{f0}\right)}$ | 20 | Dimensionless activation energy |
| $B = \dfrac{(-\Delta H) c_{A0} \gamma}{\varrho C_p T_{f0}}$ | 7.0 | Dimensionless heat of reaction |
| $\beta = \dfrac{hA}{\varrho C_p F_0}$ | 0.5 | Dimensionless transfer coefficient |
| $F_0$ | 1.0 | Nominal feed rate |
| $T_{f0}$ | 300 | Nominal feed temperature |
| $c_{A0}$ | 1.0 | Nominal feed concentration |
| $x_1 = \dfrac{c_{A0} - c_A}{c_{A0}}$ | | Outflow concentration |
| $x_2 = \dfrac{T - T_{f0}}{T_{f0}}$ | | Outflow temperature |
| $m_1 = \dfrac{T_c - T_{f0}}{T_{f0}}$ | $T_c \in [250, 350]$ | Coolant temperature |
| $m_2 = \dfrac{F - F_0}{F_0}$ | $T_c \in [0.5, 1.5]$ | Input feed rate |
| $d_1 = \dfrac{T_f - T_{f0}}{T_{f0}}$ | $T_f \in [295, 305]$ | Feed temperature |
| $d_2 = \dfrac{c_{A0} - c_{Af}}{c_{A0}}$ | $c_{Af} \in [0.9, 1.1]$ | Feed concentration |
| | Scaled inputs or outputs | |
| $y_1 = \dfrac{c_A - 0.755}{0.65}$ | $y_1 \in [-1, 1]$ | Outflow concentration |
| $y_2 = \dfrac{T - 317.1}{65}$ | $y_2 \in [-1, 1]$ | Outflow temperature |
| $u_1 = \dfrac{T_c - 300}{50}$ | $u_1 \in [-1, 1]$ | Coolant temperature |
| $u_2 = \dfrac{F - 1}{0.5}$ | $u_2 \in [-1, 1]$ | Input feed rate |

Table 3: Programmed plant tests for Case 2.

| Schedules | Set-point changes | Disturbance changes | Time |
|---|---|---|---|
| Nominal (0.000, 0.000) | $\Delta_1\mathbf{y} = (0.000, -0.150)$<br>$\Delta_2\mathbf{y} = (-0.150, 0.000)$ | $\Delta_1\mathbf{d} = (0.700, 0.000)$<br>$\Delta_2\mathbf{d} = (-0.700, 0.000)$<br>$\Delta_3\mathbf{d} = (0.000, 0.700)$<br>$\Delta_4\mathbf{d} = (0.000, -0.700)$ | @Time = (50, 100)<br>@Time = (150, 200)<br>@Time = (250, 300)<br>@Time = (350, 400) |
| Maximum gain (−0.303, 0.242) | $\Delta_1\mathbf{y} = (0.000, -0.150)$<br>$\Delta_2\mathbf{y} = (-0.150, 0.000)$ | $\Delta_1\mathbf{d} = (0.700, 0.000)$<br>$\Delta_2\mathbf{d} = (-0.700, 0.000)$<br>$\Delta_3\mathbf{d} = (0.000, 0.700)$<br>$\Delta_4\mathbf{d} = (0.000, -0.700)$ | @Time = (50, 100)<br>@Time = (150, 200)<br>@Time = (250, 300)<br>@Time = (350, 400) |
| Sign change (0.213, −0.245) | $\Delta_1\mathbf{y} = (0.000, -0.100)$<br>$\Delta_2\mathbf{y} = (-0.100, 0.000)$ | $\Delta_1\mathbf{d} = (0.700, 0.000)$<br>$\Delta_2\mathbf{d} = (-0.700, 0.000)$<br>$\Delta_3\mathbf{d} = (0.000, 0.700)$<br>$\Delta_4\mathbf{d} = (0.000, -0.300)$ | @Time = (50, 100)<br>@Time = (150, 200)<br>@Time = (250, 300)<br>@Time = (350, 400) |

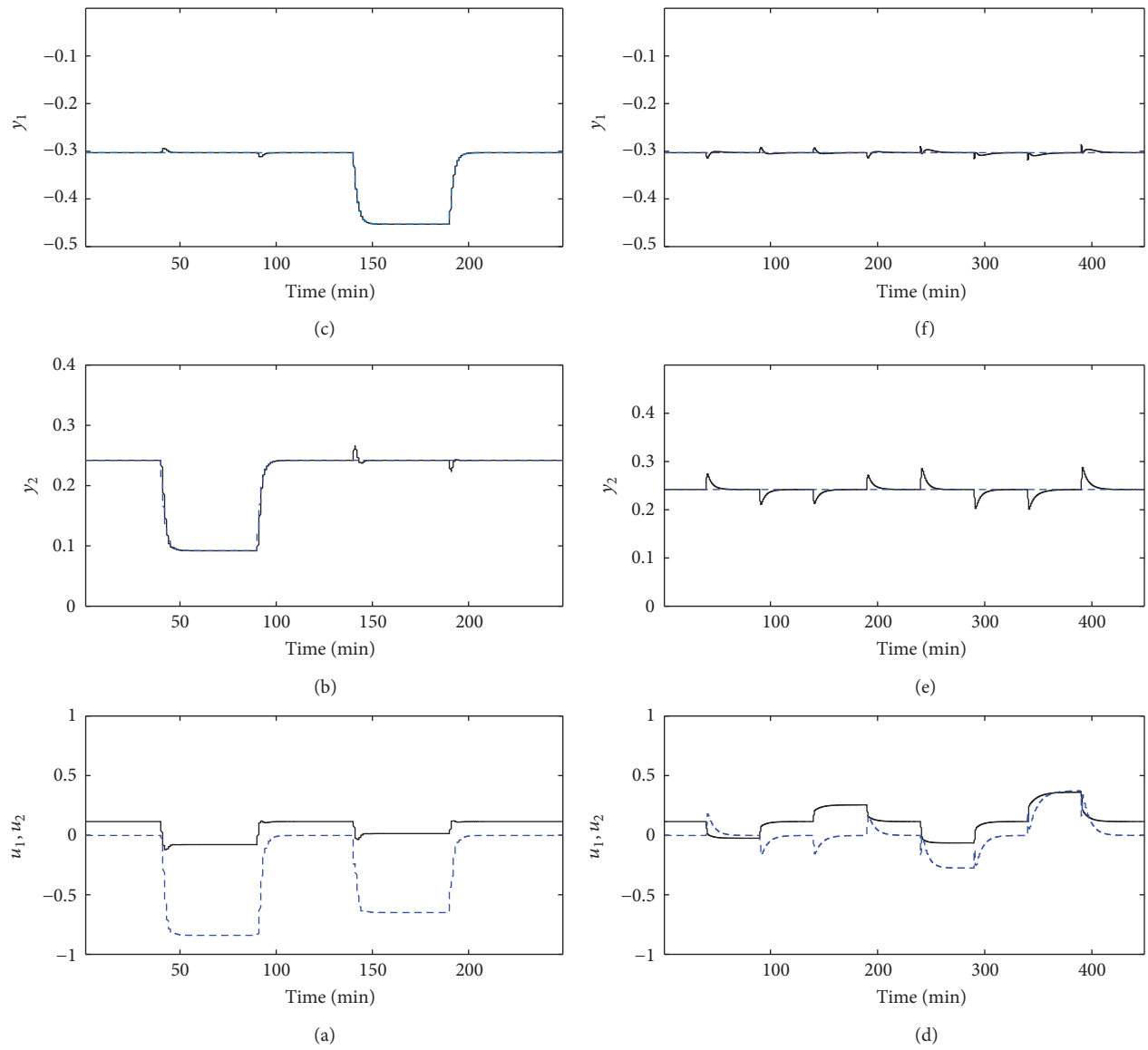


Figure 5: Performance of the gain-scheduling PI in Case 2: (a)–(c) Set-point tracking tests; (d)–(f) disturbance rejection tests. Set-points: dashed blue line; system responses: black line; $u_1$: black line; $u_2$: blue dashed line.

Table 4: Summary of ISE for Case 2.

| Conditions | Set-point tracking | Disturbance rejection |
|---|---|---|
| Nominal (0.000, 0.000) | $5.83e-2$ | $4.96e-2$ |
| Maximum gain (−0.303, 0.242) | $6.5e-3$ | $7.78e-2$ |
| Sign change (0.213, −0.245) | $1.97e-3$ | $2.08e-2$ |

neural model. These results have already been published in a previous work [13] for a different model-based controller.

## 5. Conclusion

In this paper, the use of a gain-scheduling controller strategy based on a dynamic neural network model was presented. This technique was proposed to solve problems concerning the control of highly nonlinear systems, without requiring a controller structure unusual to industrial practice. Starting from a GMC approach, the resulting algorithm is a PI controller, with an adaptive gain based on the neural model.

The performance of the developed control technique was tested in two different cases of nonisothermal CSTRs, in which an exothermic, first-order reaction takes place. The controller was applied to a wide range of set-point changes because in this way the system is forced to operate at very critical conditions, and the robustness of the control system

can be verified. The proposed control scheme was also tested in presence of disturbances in order to demonstrate the capability of the net to properly adjust gain controller values. Good results were obtained, indicating that the proposed algorithm properly adjusts the constant gain value over a wide range of operating conditions. This means that the neural model is able to describe the essential features of the process, and it captures the essential nonlinearities through an effective linear description. This characteristic enhances the adaptive controller robustness, because it performs well in the neighbourhood of the nominal operating conditions without incorporating a linear function in the neural network model [7].

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

Stefania Tronci kindly acknowledges the Fondazione Banco di Sardegna for the financial support.

## References

[1] M. Mulas, S. Tronci, F. Corona et al., "Predictive control of an activated sludge process: an application to the Viikinmäki wastewater treatment plant," *Journal of Process Control*, vol. 35, pp. 89–100, 2015.

[2] I. Machón-González and H. López-García, "Feedforward nonlinear control using neural gas network," *Complexity*, vol. 2017, Article ID 3125073, 11 pages, 2017.

[3] G. Cogoni, S. Tronci, R. Baratti, and J. A. Romagnoli, "Controllability of semibatch nonisothermal antisolvent crystallization processes," *Industrial & Engineering Chemistry Research*, vol. 53, no. 17, pp. 7056–7065, 2014.

[4] J. Na, X. Ren, C. Shang, and Y. Guo, "Adaptive neural network predictive control for nonlinear pure feedback systems with input delay," *Journal of Process Control*, vol. 22, no. 1, pp. 194–206, 2012.

[5] P. L. Lee and G. R. Sullivan, "Generic model control (GMC)," *Computers & Chemical Engineering*, vol. 12, no. 6, pp. 573–580, 1988.

[6] B. J. Cott and S. Macchietto, "Temperature control of exothermic batch reactors using generic model control," *Industrial & Engineering Chemistry Research*, vol. 28, no. 8, pp. 1177–1184, 1989.

[7] T. D. Knapp, H. M. Budman, and G. Broderick, "Adaptive control of a CSTR with a neural network model," *Journal of Process Control*, vol. 11, no. 1, pp. 53–68, 2001.

[8] A. Vega, F. Diez, and J. M. Alvarez, "Programmed cooling control of a batch crystallizer," *Computers & Chemical Engineering*, vol. 19, pp. 471–476, 1995.

[9] E. E. Ekpo and I. M. Mujtaba, "Evaluation of neural networks-based controllers in batch polymerisation of methyl methacrylate," *Neurocomputing*, vol. 71, no. 7–9, pp. 1401–1412, 2008.

[10] R. Kamesh, P. S. Reddy, and K. Y. Rani, "Comparative study of different cascade control configurations for a multiproduct semibatch polymerization reactor," *Industrial & Engineering Chemistry Research*, vol. 53, no. 38, pp. 14735–14754, 2014.

[11] S. M. Alsadaie and I. M. Mujtaba, "Generic model control (GMC) in multistage flash (MSF) desalination," *Journal of Process Control*, vol. 44, pp. 92–105, 2016.

[12] C. Foscoliano, S. Del Vigo, M. Mulas, and S. Tronci, "Predictive control of an activated sludge process for long term operation," *Chemical Engineering Journal*, vol. 304, pp. 1031–1044, 2016.

[13] R. Baratti, A. Servida, and S. Tronci, "Neural DMC control strategy for a CSTR in presence of noise," in *IFAC Symposium Dycops-6*, G. Stephanopoulos, Ed., vol. 34, pp. 680–694, 6th edition, 2001.

[14] S. S. Ge, C. C. Hang, and T. Zhang, "Nonlinear adaptive control using neural networks and its application to CSTR systems," *Journal of Process Control*, vol. 9, no. 4, pp. 313–323, 1999.

[15] G. M. Scott and W. H. Ray, "Creating efficient nonlinear neural network process models that allow model interpretation," *Journal of Process Control*, vol. 3, no. 3, pp. 163–178, 1993.

[16] J. C. Principe, N. R. Euliano, and W. C. Lefebvre, *Neural and Adaptive Systems: Fundamentals through Simulations*, vol. 672, Wiley, New York, NY, USA, 2000.

[17] B. A. Ogunnaike and W. H. Ray, *Process Dynamics, Modeling and Control*, Oxford University Press, 1994.

[18] G. Lightbody and G. W. Irwin, "Direct neural model reference adaptive control," *IEE Proceedings Control Theory and Applications*, vol. 142, no. 1, pp. 31–43, 1995.