Journal of Process Control 23 (2013) 968–979

Contents lists available at SciVerse ScienceDirect

Journal of Process Control

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

An offset-free neural controller based on a non-extrapolating scheme for approximating the inverse process dynamics

Alex Alexandridis[a,*], Marios Stogiannos[a], Alexandra Kyriou[b], Haralambos Sarimveis[b]

[a] Department of Electronics, Technological Educational Institute of Athens, Agiou Spiridonos, Aigaleo 12210, Greece
[b] School of Chemical Engineering, National Technical University of Athens, Greece

ARTICLE INFO

Article history:
Received 15 January 2013
Received in revised form 28 April 2013
Accepted 29 April 2013

Keywords:
Intelligent control
Neuro-control
Neural networks
Inverse dynamics
Radial basis functions
Offset-free control

ABSTRACT

This work presents a novel control scheme based on approximating the inverse process dynamics with a radial basis function (RBF) neural network model, trained with the fuzzy means algorithm. The produced RBF network constitutes an inverse model of the process, which can be applied as an explicit control law. In order to avoid extrapolation in the RBF model predictions, a concept borrowed from chemometrics, namely the applicability domain, is incorporated to the proposed framework. Moreover, an error correction term is added, allowing the inverse neural controller to account for modeling errors and process uncertainty and eliminate offset. The proposed approach is applied to the control of a nonlinear Continuous Stirred Tank Reactor (CSTR) exhibiting multiple equilibrium points, including an unstable one. A comparison with other control schemes on various tests, including set-point tracking, unmeasured disturbance rejection and process uncertainty highlights the advantages of the proposed controller.

1. Introduction

Computational intelligence has been used extensively during the last two decades for designing novel automatic control schemes, especially in cases where conventional methods fail to develop appropriate dynamic models of the process and/or compute the control law [1]. In such situations, the Artificial Neural Network (ANN) technology offers an efficient alternative approach for training dynamic models of nonlinear systems using experimental information from the process [2,3] and for synthesizing effective nonlinear controllers [4,5]. The prospect of using ANN methodologies in chemical process control was reviewed by Hussain [6].

There are three major directions in the design of discrete time control systems based on ANN techniques, which can be complementary, namely model predictive controllers (MPCs), adaptive control techniques and inverse model-based control methodologies [6,7]. In MPC, neural network methodologies are applied on historical dynamic input–output data to construct the nonlinear dynamic predictive model of the process [8–10]. Results have also been reported on the development of reduced-order models, simplifying the formulation of the optimization problem which is solved in real time [11]. Adaptive MPC methods based on ANNs extend the applicability of standard MPC systems to the control of time-varying dynamics, by adding online adaptation capabilities to the nonlinear models [12–15].

The main obstacle in the application of nonlinear MPC is that the nonlinear optimization problem that is formulated at each discrete time instant must be solved in real time [16]; nevertheless, the available time between two subsequent time instants may not be sufficient even for satisfying the requirements of suboptimal MPC [17]. To overcome this problem and extend the applicability of MPC to systems with fast dynamics, the concept of explicit MPC was coined a decade ago [18]. Under this approach, the computational burden of MPC is moved offline, by pre-computing the optimal MPC control law as a function of the current state. Today, concrete theory exists that guarantees the exact solution of explicit MPC for linear systems [19], but only approximate explicit MPC methods have been developed for nonlinear systems [20,21]. Explicit MPC actually partitions the state space in regions and in each region a different control law is defined. However, explicit MPC suffers from the curse of dimensionality: especially for problems of high dimensionality, the process of finding the correct region where the current state belongs can be more computationally demanding than solving the online optimization problem. Improving the computational speed is the most important research issue today in the MPC literature [16], but the development of customized fast optimization algorithms [22,23] is restricted to linear systems, or systems iteratively linearized online [24,25]. Another possibility is to completely avoid the solution of the optimization problem,

* Corresponding author. Tel.: +30 2105385358.
E-mail address: alexx@teiath.gr (A. Alexandridis).

0959-1524/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.jprocont.2013.04.008

substituting it with a neural network approximator, trained off-line to minimize a control-relevant cost function [8].

The development of inverse model-based neural control methodologies [6,7] has evolved along two directions: (a) an extension of the linear Internal Model Control (IMC) method and (b) the direct inverse control method.

Following the IMC approach, an ANN dynamic model runs in parallel with the actual process and the difference (error) between the two outputs, which is due to model uncertainties and/or external disturbances, is subtracted from the set-point before being fed to the inverse model [26–28]. The inverse model is realized either by numerically inverting the neural network at each interval using an approach like Newton's method, or by training a second neural network using the inputs of the process as outputs and vice versa. Numerical inversion is computationally intensive and time-consuming, rather sensitive to initial estimates and may not necessarily give the global optimum solution [29,30]. The incorporation of a second neural network model on the other hand introduces extra complexity in the overall control scheme.

The idea behind the direct inverse control approach is to train an ANN as the inverse of the plant, so as to cancel the system dynamics and make the plant follow the reference input. Therefore, using this approach, a single ANN model is developed that acts as the control law, computing the values of the manipulated variables that are used as inputs to the process at each discrete time instant. Past values of input and output (or state) variables of the system along with the desired set-points constitute the input vector to the ANN model [31]. Measured disturbances have also been included in the input vector in direct inverse control formulations [32]. Noticing that MPC acquires the form of an inverse model of the system in the event of a predictive horizon being equal to one, the term "predictive inverse neuro-control" was recently suggested as an alternative to direct inverse control [33].

As far as computational load is concerned, the direct inverse control approach is very efficient as it just requires the evaluation of a nonlinear function. Although most direct inverse control formulations proposed in the literature are not able to account for model uncertainties and unmeasured external disturbances [34–37], performance of the inverse neural controller can be improved by selecting appropriate excitation signals [31,38]. The additional requirement of offset-free control can be achieved by adapting online the inverse model parameters [39].

In this work we introduce a new direct inverse neural control method, which retains the low computational complexity of the standard inverse-based approach and additionally is able to produce zero-offset even in the events of unmeasured external disturbances or uncertainties in plant dynamics, without the need to adapt the model. The method uses the radial basis function (RBF) neural network architecture [40,41] and the inverse neural controller is trained using the fuzzy means algorithm [42] on experimental input–output examples which are collected during the dynamic operation of the system. A concept known as the applicability domain [43] is borrowed from the field of chemometrics, to avoid extrapolation in the predictions of the inverse neural controller. The extrapolation phenomenon results to unreliable predictions when the model is used in regions of the input space that have not been covered by training data, and thus, may considerably deteriorate the controller performance.

The rest of this paper is structured as follows: In the next section, the RBF neural network architecture is presented, followed by a brief description of the network training methodology used in this work. Section 3 introduces the proposed direct inverse control methodology, describing in details the incorporation of the applicability domain criterion in the controller design, and the addition of the error correction term. In Section 4, the efficiency of the proposed control scheme is illustrated by applying the method to the full range control of a nonlinear CSTR containing two stable and one unstable equilibrium points and comparing it with alternative controllers. Finally, the paper ends with the concluding section.

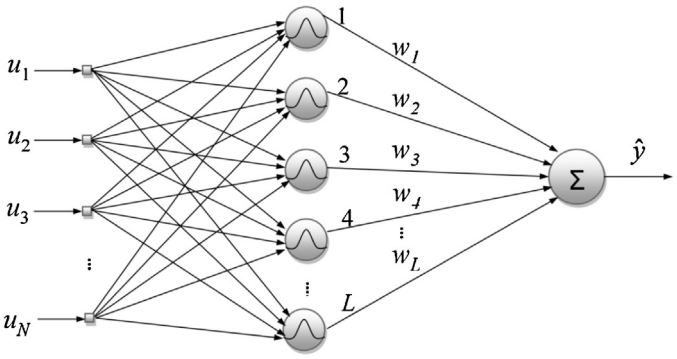

Fig. 1. Typical structure of an RBF network.

2. Radial basis function networks

An RBF network can be considered as a special three layer neural network, which is linear with respect to the output parameters after fixing all the radial basis function centers and nonlinearities in the hidden layer. The typical structure of an RBF network is shown in Fig. 1. The input layer distributes the N input variables to the L nodes of the hidden layer. Each node in the hidden layer is associated with a center, equal in dimension with the number of input variables. Thus, the hidden layer performs a nonlinear transformation and maps the input space onto a new higher dimensional space. The activity $\mu_l(\mathbf{u}_k)$ of the lth node is the Euclidean norm of the difference between the kth input vector and the node center and is given by:

$$\mu_l(\mathbf{u}_k) = \left\|\mathbf{u}_k - \hat{\mathbf{u}}_l\right\| = \sqrt{\sum_{i=1}^{N}(u_{k,i} - \hat{u}_{l,i})^2}, \quad k = 1, 2, \ldots, K \tag{1}$$

where K is the total number of data, $\mathbf{u}_k^T = \left[u_{k,1},\ u_{k,2},\ \ldots,\ u_{k,N}\right]$ is the input vector and $\hat{\mathbf{u}}_l^T = [\hat{u}_{l,1}, \hat{u}_{l,2}, \ldots, \hat{u}_{l,N}]$ is the center of the lth node.

The output function of the node is a radially symmetric function. This work employs the thin plate spline function:

$$h(\mu) = \mu^2 \log(\mu) \tag{2}$$

The final output $\hat{t}_k$ of the RBF network for the kth data point is produced by a linear combination of the hidden node responses, after adjusting the weights of the network appropriately:

$$\hat{t}_k = \sum_{l=1}^{L} w_l h(\mu_l(\mathbf{u}_k)),\ k = 1, 2, \ldots, K \tag{3}$$

where w_l stands for the synaptic weight of the lth node.

2.1. The fuzzy means algorithm

The fuzzy means (FM) algorithm has been introduced a decade ago in order to replace the k-means algorithm in the selection of the hidden layer nodes [40,42]. The FM algorithm presents several advantages over the typical approach, including faster computational times and automatic determination of the size of the network, and has been used successfully in a number of control-related applications [12,44–46]. Recently a non-symmetric version of the algorithm with improved prediction abilities has been introduced [47]. A brief discussion about the FM algorithm is given

below, while the interested reader is referred to the original publications.

Consider a system with N normalized input variables u_i, where $i = 1, \ldots, N$. The domain of each input variable is partitioned into an equal number of one-dimensional triangular fuzzy sets, c. Each fuzzy set can be written as:

$$A_{i,j} = \{a_{i,j}, \delta\alpha\}, \quad i = 1, \ldots, N, \quad j = 1, \ldots, c \tag{4}$$

where $a_{i,j}$ is the center element of fuzzy set $A_{i,j}$ and $\delta\alpha$ is half of the respective width (due to the symmetric partition all the widths are equal). This partitioning technique creates a total of c^N multi-dimensional fuzzy subspaces $\mathbf{A}^l$, where $l = 1, \ldots, c^N$. Each multi-dimensional fuzzy subspace is generated by combining N one-dimensional fuzzy sets, one for each input direction. The center vector $\mathbf{a}^l$ and the side vector $\boldsymbol{\delta\alpha}$ of each fuzzy subspace can be defined as follows:

$$\mathbf{A}^l = \{\mathbf{a}^l, \boldsymbol{\delta\alpha}\} = \left\{ [a^l_{1,j_1}, a^l_{2,j_2}, \ldots, a^l_{N,j_N}], [\underbrace{\delta\alpha, \delta\alpha, \ldots, \delta\alpha}_{N}] \right\}, \quad l = 1, \ldots, c^N \tag{5}$$

where a^l_{i,j_i} is the center element of the one-dimensional fuzzy set A_{i,j_i} that has been assigned to input i. Each one of the produced fuzzy subspaces is a candidate for becoming an RBF center, but only a few of those will be finally selected. The selection is based on the idea of the multidimensional membership function $\mu_{\mathbf{A}^l}(\mathbf{u}_k)$ of an input vector $\mathbf{u}_k$ to a fuzzy subspace $\mathbf{A}^l$ which is given by Nie [48]:

$$\mu_{\mathbf{A}^l}(\mathbf{u}_k) = \begin{cases} 1 - r_l(\mathbf{u}_k), & \text{if } r_l(\mathbf{u}_k) \le 1 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $r_l(\mathbf{u}_k)$ is the Euclidean relative distance between $\mathbf{A}^l$ and the input data vector $\mathbf{u}_k$:

$$r_l(\mathbf{u}_k) = \frac{\sqrt{\sum_{i=1}^{N} (a^l_{i,j_i} - u_{i,k})^2}}{\delta\alpha\sqrt{N}} \tag{7}$$

Eq. (7) defines a hyper-sphere on the input space with radius equal to $\delta\alpha\sqrt{N}$. The objective of the training algorithm is to select a subset of fuzzy subspaces as RBF centers, so that all the training data are covered by at least one hyper-sphere. Expressing this requirement in terms of Eq. (6), the subset of fuzzy subspaces is selected so that there is at least one fuzzy subspace that assigns a nonzero multidimensional degree to each input training vector.

The synaptic weights are calculated using linear regression of the hidden layer outputs to the real measured outputs (target values). The regression problem can be trivially solved using linear least squares in matrix form:

$$\mathbf{W} = \mathbf{T}^T\mathbf{Z}(\mathbf{Z}^T\mathbf{Z})^{-1} \tag{8}$$

where $\mathbf{W}$, $\mathbf{Z}$ and $\mathbf{T}$ are matrices containing the synaptic weights, hidden layer outputs and target values, respectively.

3. Design of the inverse neural controller

3.1. Inverse dynamic model used for designing a discrete controller

Let us assume that the process under control is a single input–single output (SISO) process with N state variables. A discrete dynamic model of the process correlates the next value of the output (controlled) variable with the current values of the state variables and the input (manipulated) variable:

$$y(k+1) = f(\mathbf{x}(k), v(k)) \tag{9}$$

where y is the controlled variable, $\mathbf{x}$ is the state vector and v is the manipulated variable. We can assume, without loss of generality, that f is a nonlinear function. The discrete inverse dynamic model of the process can be formulated as follows:

$$v(k) = g(\mathbf{x}(k), y(k+1)) \tag{10}$$

where g is the inverse dynamic function of the process. Substituting the next value of the controlled variable $y(k+1)$ in Eq. (10) with the current value of the set-point $\omega(k)$, essentially provides a control law which can be used for calculating which value of the manipulated variable should be applied at any discrete time instant, in order to drive the process from its current state, to the set-point:

$$v(k) = g(\mathbf{x}(k), \omega(k)) \tag{11}$$

Under the assumptions that all state variables are measured and a sufficient number of training examples is available, the FM algorithm presented in section 2.1 can be used to approximate the unknown function g.

The neural network approximator of the inverse dynamics can then act as a controller, receiving at each discrete time instant the desired set-point value and the values of the state variables and producing the control command, i.e. the value of the manipulated variable, which will be applied to the system during the subsequent time period:

$$v(k) = \mathrm{RBF}(\mathbf{x}(k), \omega(k)) \tag{12}$$

where RBF is the nonlinear function implemented by the RBF network, calculated through Eq. (3). The operation of such a controller is depicted in Fig. 2.

3.2. Extrapolation and applicability domain

It is well known that neural network models – as well as other black-box modeling techniques – have a limited ability for generalization. This comes as a direct result of the inadequacy of such models for providing reliable predictions in regions of the input space which have not been covered sufficiently in the training dataset, i.e. the inability to successfully perform *extrapolation*. Notice that in this context, extrapolation does not refer only to exceeding the minimum and maximum values of the training data in each input dimension; extrapolation may well occur within training limits when there are no training data around the point of interest. Fig. 3 presents an example in a two-dimensional input space, depicting a set of training data, and the areas where extrapolation occurs.

The performance of a neural controller may deteriorate significantly, due to the poor extrapolation capabilities of the incorporated neural network model. Unfortunately, the discrete inverse neural controller adhering to Eq. (12) is prone to extrapolation as well. The reasons for this are twofold: firstly, there are practical limitations in achieving an adequate process excitation level in order to obtain training data that cover sufficiently all feasible combinations of the state vector components $\begin{bmatrix} x_1(k) & x_2(k) & \ldots & x_N(k) \end{bmatrix}$. The second reason can be explained by examining the input vector $[\mathbf{x}(k)\ \omega(k)]$ that is provided to the neural controller at each particular time instant k, which includes not only the state vector, but also the set-point value $\omega(k)$. Depending on the relative values of $\mathbf{x}(k)$ and $\omega(k)$ and the sampling time of the controller, it may be infeasible to move the process from its current state $\mathbf{x}(k)$ to the desired set-point value $\omega(k)$, within a single discrete time period. In such a case, the neural controller will inevitably be asked to extrapolate, as it is obvious that no similar examples have been used during the training phase of the neural network. This situation would result to the calculation of an unreliable value for the manipulated variable, obviously deteriorating the performance of the controller.

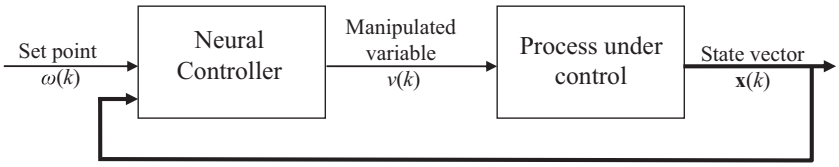

Fig. 2. A neural controller based on an inverse process model.

In order to avoid the undesirable effects of extrapolation in the prediction of the manipulated variable, we introduce to the neural controller design problem a concept widely used in chemometrics, known as the applicability domain (AD) of the model, i.e. methodologies that define in a quantitative manner the subregions of the input space that have been sufficiently covered by training examples and thus help to produce reliable neural network predictions. A popular method for assessing the AD of neural network models computes the leverage value, which is proportional to the Hotelling T^2 statistic and to the Mahalanobis distance [43]. The leverage $h(k)$ of an input vector $\mathbf{u}(k)$ in the input space is given by:

$$h(k) = \mathbf{u}^T(k) \cdot (\mathbf{U} \cdot \mathbf{U}^T)^{-1} \cdot \mathbf{u}(k) \tag{13}$$

where $\mathbf{U}$ is a matrix containing all the input training examples.

The input $\mathbf{u}(k)$ is considered to be inside the AD when the following condition is true:

$$h(k) \leq 3\frac{N+1}{K} \tag{14}$$

When an input lies outside of the AD, then the corresponding prediction is the result of extrapolation.

3.3. Incorporation of the AD criterion in the inverse controller design

The notion of AD is incorporated in the proposed methodology to avoid extrapolation at each discrete time instant, by formulating a suitable input vector to the neural controller. More precisely, the AD concept is used to define upper and lower bounds on the value of $\omega(k)$ that guarantee interpolation. This is accomplished by formulating at each discrete time step during the operation of the neural controller, the following equation:

$$\left[\mathbf{x}(k) \quad \omega(k)\right]^T \cdot (\mathbf{U} \cdot \mathbf{U}^T)^{-1} \cdot [\,\mathbf{x}(k) \quad \omega(k)\,] = 3\frac{N+1}{K} \tag{15}$$

Notice that the state vector $\mathbf{x}(k)$ is known at each time instant, therefore Eq. (15) is a second-order equation with respect to $\omega(k)$. Its two solutions $\omega_{\min}(k)$ and $\omega_{\max}(k)$ actually define the lower and upper bounds on the value of $\omega(k)$ that guarantee that extrapolation is avoided. In cases where the current set-point $\omega(k)$ lies outside of these bounds, it is substituted by the closest bound, thus keeping the input vector to the neural controller within the AD:

$$\omega_{\mathrm{RE}}(k) = \begin{cases} \omega_{\min}(k), & \text{if } \omega(k) < \omega_{\min}(k) \\ \omega_{\max}(k), & \text{if } \omega(k) > \omega_{\max}(k) \\ \omega(k), & \text{if } \omega(k) \in [\omega_{\min}(k), \omega_{\max}(k)] \end{cases} \tag{16}$$

where $\omega_{\mathrm{RE}}(k)$ is the requested set-point value which is given as input to the RBF network to produce the control law:

$$v(k) = \mathrm{RBF}(\mathbf{x}(k),\ \omega_{\mathrm{RE}}(k)), \tag{17}$$

In order to account for inaccuracies in the training data, the limits defining the AD can be tightened. This is accomplished by dividing the length of the line segment between $\omega_{\min}(k)$ and $\omega_{\max}(k)$ by a narrowing parameter r, where $r > 1$. Thus, a narrower AD is produced whose bounds are given by:

$$\omega'_{\min}(k) = \frac{(r+1)\omega_{\min}(k) + (r-1)\omega_{\max}(k)}{2r} \tag{18}$$

$$\omega'_{\max}(k) = \frac{(r+1)\omega_{\max}(k) + (r-1)\omega_{\min}(k)}{2r} \tag{19}$$

The stricter bounds $\omega'_{\min}(k)$ and $\omega'_{\max}(k)$ are now utilized to calculate the requested set-point value:

$$\omega_{\mathrm{RE}}(k) = \begin{cases} \omega'_{\min}(k), & \text{if } \omega(k) < \omega'_{\min}(k) \\ \omega'_{\max}(k), & \text{if } \omega(k) > \omega'_{\max}(k) \\ \omega(k), & \text{if } \omega(k) \in \left[\omega'_{\min}(k), \omega'_{\max}(k)\right] \end{cases} \tag{20}$$

while the control law is still implemented using Eq. (17). A schematic overview of the required steps to incorporate the AD criterion to the controller design is given in Fig. 4.

Introduction of the narrowing parameter r also provides a means for tuning the neural controller. Larger values of r imply that control actions will be more conservative as the requested set-point value per time step will be closer to the current value of the controlled variable. On the other hand smaller values of r will trigger more aggressive control actions, in order to traverse the increased distance to the requested set-point.

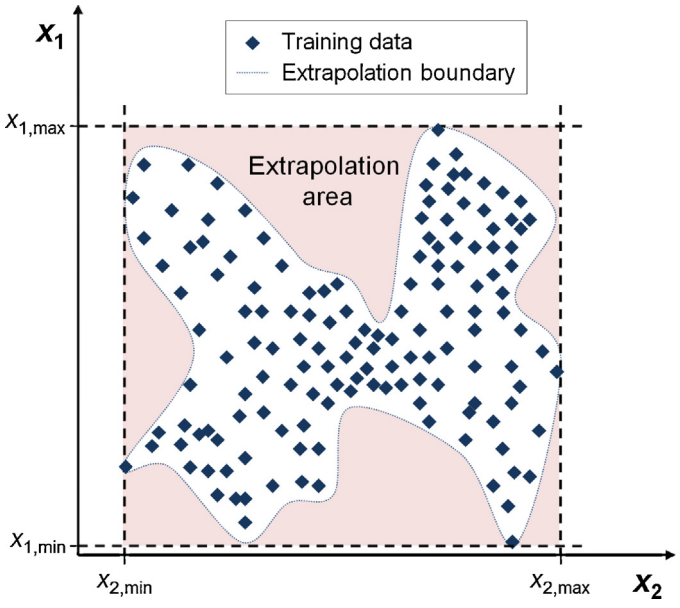

Fig. 3. A two-dimensional example, depicting a set of training data and the boundary of extrapolation.

3.4. Offset-free control

Eq. (17) describes a model for predicting which value of the manipulated variable will drive the process from its current state to the new set-point. Obviously the resulting controller is susceptible to unmeasured disturbances and modeling errors, which affect both the dynamic and static performance of the controlled

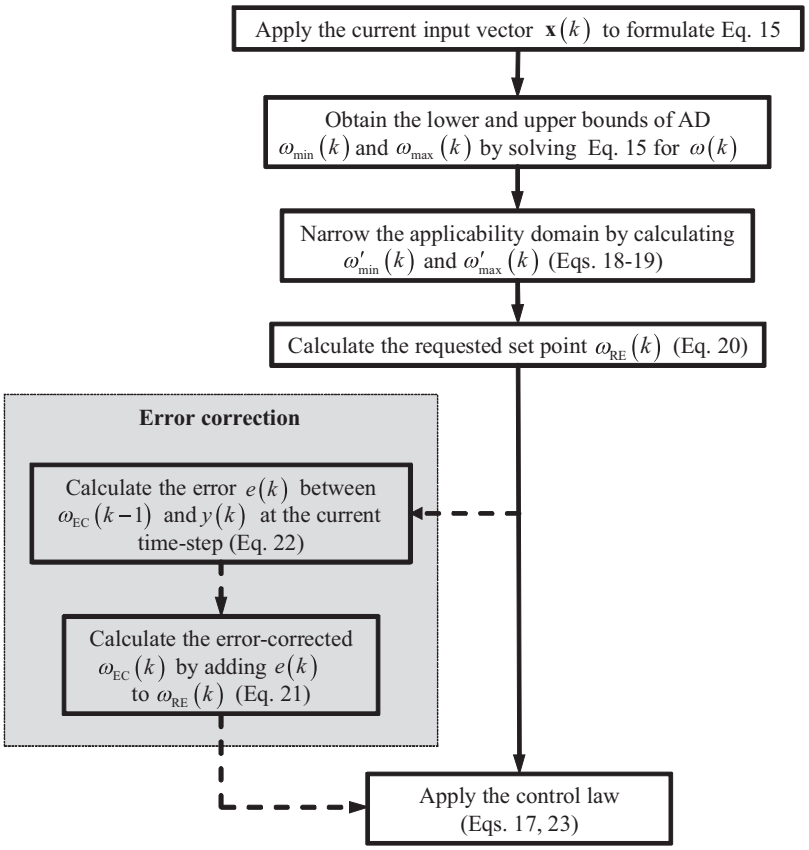

Fig. 4. Required steps for incorporating the AD criterion to the controller design. The additional steps required for incorporating an error correction term are depicted with dashed lines.

system. In order to account for such cases, we introduce an error-correction term which is added to the requested set-point value at each time step. This approach resembles a concept used in off-set free MPC, where the state vector of the predictive process model is augmented by a disturbance state, which captures the error between the model and the true plant dynamics [49,50]. The error-corrected requested set-point $\omega_{EC}(k)$ is calculated by adding the error term to the calculated requested set-point $\omega_{RE}(k)$:

$$\omega_{EC}(k) = \omega_{RE}(k) + e(k) \tag{21}$$

where $\omega_{RE}(k)$ is calculated using Eq. (20) and the error $e(k)$ is the difference between the error-corrected set-point at the previous time step and the value of the controlled variable at the current time-step:

$$e(k) = \omega_{EC}(k-1) - y(k) \tag{22}$$

The control law is now updated to include the error-corrected requested set-point:

$$v(k) = \mathrm{RBF}(\mathbf{x}(k), \omega_{EC}(k)), \tag{23}$$

Eqs. (21)–(22) imply that the error of the previous time step will remain constant during the next step, therefore the requested set-point is corrected to compensate for this error. Of course there is no guarantee that the error indeed remains constant between two consecutive time-steps; however, this is the best assumption that can be made based on the available information. Notice that as the AD of the controller gets narrower by increasing the tuning parameter r, the allowed set-point changes between two successive time steps get smaller. This results to smaller variation in the input vector, as far as the requested set-point component $\omega_{RE}(k)$ is concerned, thus strengthening the assumption that the prediction error remains constant between consecutive time steps.

Incorporation of the error-correction term to the controller design is also presented schematically in Fig. 4.

4. Case study: control of a nonlinear CSTR with multiple steady-states

4.1. CSTR simulation

This section presents an example where the proposed inverse neural controller is applied to the control of a simulated system. The simulation involves a non-isothermal CSTR where the following exothermal irreversible reaction between sodium thiosulfate and hydrogen peroxide is taking place:

$$2Na_2S_2O_3 + 4H_2O_2 \rightarrow Na_2S_3O_6 + Na_2SO_4 + 4H_2O \tag{24}$$

This process is described by the following mass and energy balances [51]:

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{F}{V}(C_{A,in} - C_A) - 2k_o \exp\left(-\frac{E}{RT}\right) C_A^2 \\ \frac{dT}{dt} &= \frac{F}{V}(T_{in} - T) + 2\frac{(-\Delta H)_R}{\rho c_p} k_o \exp\left(-\frac{E}{RT}\right) C_A^2 - \frac{UA}{V\rho c_p}(T - T_j) \end{aligned} \tag{25}$$

where V is the volume of the CSTR; $(\Delta H)_R$ is the heat of the reaction; $-E/R$, k_o, c_p, ρ are constants of the reaction and the reactants; F is the flow rate into the reactor; $C_{A,in}$ is the inlet concentration of the reactant $Na_2S_2O_3$; C_A is the concentration of $Na_2S_2O_3$ inside the reactor; T_{in} is the inlet temperature; T is the temperature inside the

Table 1
Process parameter values for the CSTR.

Process parameter	Value
V	100 l
UA	20,000 J/s K
ρ	1000 g/l
C_p	4.2 J/g K
$-(\Delta H)_R$	596,619 J/mol
k_o	6.85E+11 l/s mol
E	76534.704 J/mol
R	8.314 J/mol K
F	20 l/s
T_{in}	275 K
$C_{A,in}$	1 mol/l

reactor; T_j is the temperature of the coolant and UA is the overall heat-transfer coefficient between the cooling coil and the reactor contents. The values of the process parameters are shown in Table 1.

It is well known that this type of CSTR has normally three steady-state points, an upper and a lower one which are stable, and a middle one which is unstable. Though it is relatively easy to control the CSTR around the individual upper or lower steady-state point, controlling the CSTR over the whole operating range which includes the unstable steady-state point is a rather challenging task.

The CSTR was simulated by solving the system of ODEs in Eq. (25). The objective was to apply the proposed neural controller in order to control the concentration C_A inside the CSTR using the temperature of the coolant T_j as the manipulated variable, assuming that both process states (C_A, T) can be measured in real time.

4.2. Inverse model training

In the case of the CSTR, the inverse process model has the following form:

$$T_j(k) = \mathrm{RBF}(C_A(k), T(k), C_A(k+1)) \tag{26}$$

Given the full state vector $[C_A(k)\, T(k)]$ at the current time instant, the model predicts the value of the manipulated variable $T_j(k)$ that must be applied to the system, so that the concentration at the next time instant will be equal to $C_A(k+1)$. In order to generate data for training the inverse RBF model, random changes from a uniform distribution were applied to the coolant temperature T_j every 0.5 s, within the limits 0–500. Using the described configuration, 60,000 data points were collected from the CSTR. The data points were split into a training dataset of 42,000 data points and a validation one of 18,000 data points. After collecting the data, an exhaustive search was performed testing partitions ranging from 4 to 100 fuzzy sets. The best network, which was selected based on the Root Mean Square Error (RMSE) value on the validation dataset, was the one employing a partition of 88 fuzzy sets, resulting to 726 centers. For this network, the RMSE and R^2 pointers on the validation dataset were equal to 18.76 and 0.965, respectively. Fig. 5 depicts the real values for the manipulated variable T_j together with the neural network predictions for a portion of the validation data.

4.3. Set-point tracking

In order to evaluate the performance of the inverse neural controller in set-point tracking, a test comprising a series of set-point changes was improvised, aiming to cover the whole operating region of the CSTR.

For comparison purposes, the test included a simple inverse neural controller (INC), an inverse neural controller taking into account the AD (INCAD) and a PID controller. In order to tune the latter, the CSTR was firstly linearized around the unstable steady-state point, using the state equations (Eq. (25)). The PID controller

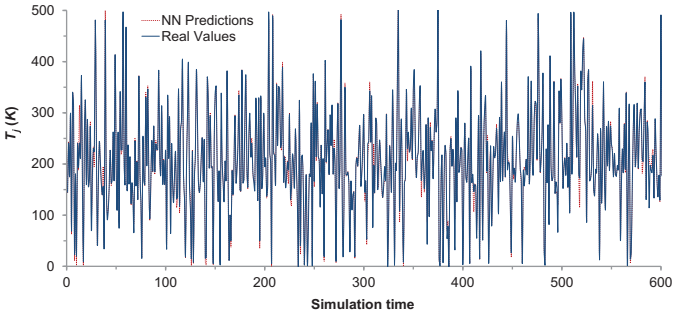

Fig. 5. Real values for the manipulated variable T_j, together with neural network predictions for a portion of the validation data.

was then tuned by applying the IMC procedure, where a first-order filter was used and the tuning parameter λ was optimized in order to minimize the Mean Absolute Error (MAE) criterion for the actual closed-loop response [52]. Mean Absolute Error was calculated as follows:

$$MAE = \frac{\sum_{K=1}^{K_t} \left|\omega(k) - C_A(k)\right|}{K_t} \tag{27}$$

where K_t stands for the total number of time-steps in the simulation.

A value of $r = 2.5$ was selected for the narrowing parameter of the INCAD controller. As it will be shown later, this value produces the lowest MAE compared to different values of r, providing a balance between aggressive and conservative behavior. Fig. 6a shows the responses for the three controllers, together with the set-point changes, whereas the control actions are depicted in Fig. 6b. The values of MAE for the three controllers are presented in Table 2. It can be seen that the INC controller manages to control the CSTR around the upper and lower stable steady-state points but fails to do so around the middle unstable steady-state point. The INC control structure was rendered able to control the process around the unstable steady-state only after decreasing the sampling period by one order of magnitude; however this may be impractical in real-world implementations, especially as far as concentration measurement is concerned. The INCAD controller on the other hand exploits the AD information, thus succeeding in driving the CSTR successfully through the whole operating region, without the need to decrease the sampling time. Between these two, the INC controller appears to have a slightly faster response; this is expected, as taking into account the AD forces the INCAD controller to make smaller steps toward the set-point. However, the superiority of INCAD is obvious, as not only does it reach the unstable steady-state, but also provides smaller overshoot values and settling times for all set-point changes. The PID controller on the other hand, is significantly slower compared to the other two controllers, a fact which is reflected on its MAE value. However, any attempt to tune the PID for faster response times, results to instability. It should be noted that, if given more time between the step point changes, the PID finally succeeds in reaching the set-point value, even for the unstable steady-state point. However,

Table 2
Set-point tracking problem: Values for Mean Absolute Error (MAE) for the three controllers.

Controller	MAE
INCAD ($r = 2.5$)	0.079
INC	0.2
PID	0.214

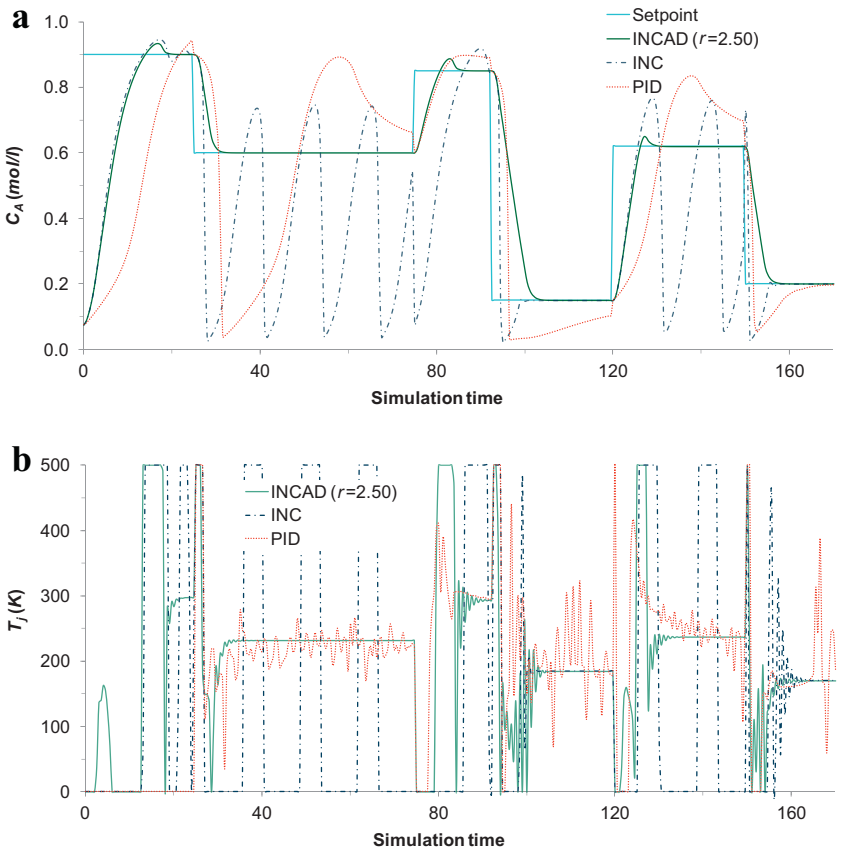

Fig. 6. Set-point tracking problem: (a) responses for the three controllers, together with the set-point changes and (b) control actions for the three controllers.

the INCAD controller is still vastly superior in terms of performance characteristics (overshoot, settling time and response time) for all set-point changes. Fig. 7 depicts the response of the INCAD and PID controllers applying the same set-point changes, but allowing the set-point to be modified only after the PID response reaches a steady-state value.

In order to assess the impact of the narrowing parameter r to the performance of the INCAD controller, we tested four different values of r, namely 1.5, 2.5, 3.5 and 4.5. The four responses and the control actions are depicted in Fig. 8a and b, respectively, whereas the values for the MAE pointer are shown in Table 3. It can be seen that increasing the narrowing parameter, alters the behavior of the controller from aggressive with fast but oscillatory response to conservative with no oscillations but slower response. Thus the choice of the value for r actually depends on the desired characteristics of the closed loop response. An important advantage of this approach compared to other methods, such as MPC, is

Table 3
Impact of narrowing parameter r: Values for Mean Absolute Error (MAE) for the four controllers.

Controller	MAE
INCAD (r=1.5)	0.134
INCAD (r=2.5)	0.079
INCAD (r=3.5)	0.096
INCAD (r=4.5)	0.116

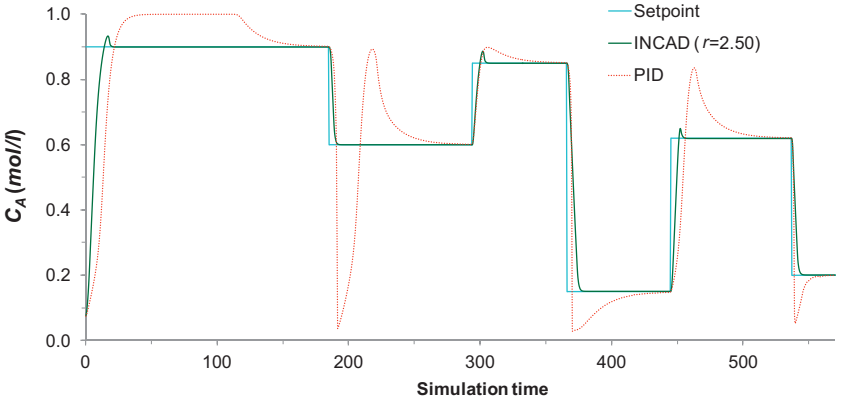

Fig. 7. Set-point tracking problem: responses for the PID and INCAD controllers, together with the set-point changes. The set-point is allowed to be modified only after the PID response reaches a steady-state value.

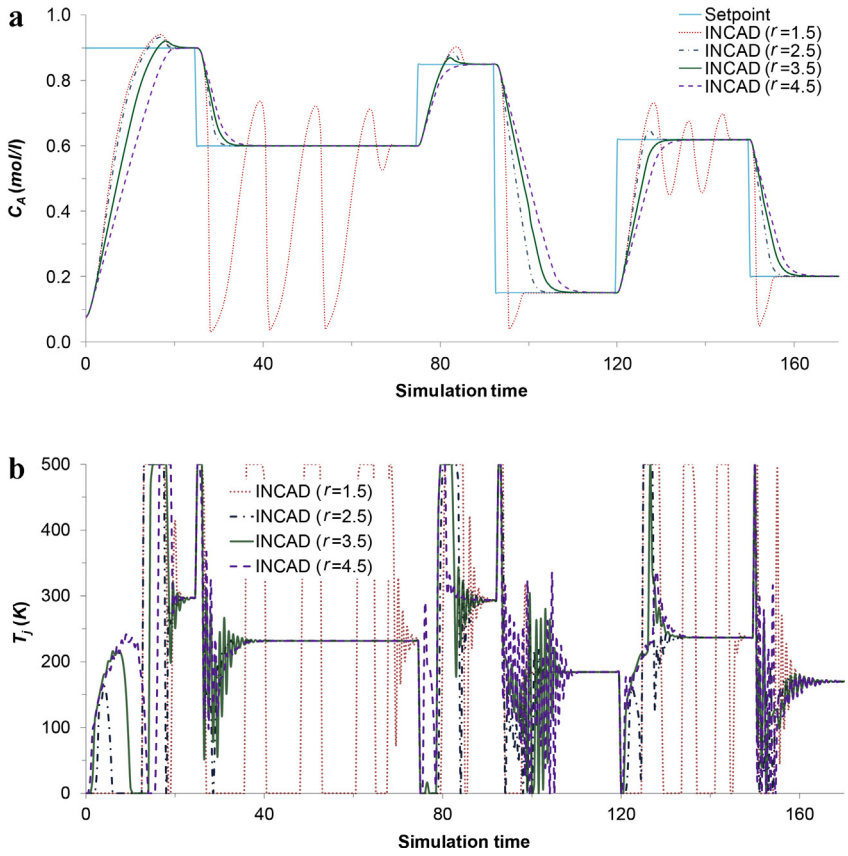

Fig. 8. Impact of narrowing parameter r: (a) responses for the four controllers, together with the set-point changes and (b) control actions for the four controllers.

that tuning of the controller boils down to choosing one value for a single parameter.

The inverse neural controller taking into account the applicability domain with error correction (INCADEC) was also tested on the same case study. As already discussed in Section 3, inclusion of the error correction term requires a more conservative choice for the narrowing parameter. Table 4 depicts results for the INCAD and INCADEC controllers, as far as MAE, the maximum absolute steady-state error (MaxASSE) and the mean absolute steady-state error (MASSE) for all set-point changes are concerned. It can be seen that the error correction term helps the INCADEC controller to completely eliminate steady-state error, thus achieving offset-free control.

4.4. Unmeasured disturbance

In this test, the performance of the controllers is tested when an unmeasured disturbance enters the process. To be more specific, a disturbance to the inlet temperature T_{in} from 275 K to 305 K was introduced and the set-point value was fixed at the unstable steady-state point. The responses, control actions and MAE values for the INC, INCAD, INCADEC and PID controllers are shown in Fig. 9a and b and Table 5, respectively. It can be seen that the INC controller becomes unstable with an oscillatory response, while the INCAD controller manages to keep the CSTR near the unstable operating region but obviously fails to reject the disturbance. This was expected since these two controllers have no means of compensating for any change that may occur in the process after the training data collection stage. The INCADEC controller on the other hand, manages to return the controlled variable to the desired set-point value, despite the severity of the disturbance. In order to achieve this, however, an even more conservative choice for the narrowing parameter is needed ($r = 10$). Still, the INCADEC controller is considerably faster in rejecting the disturbance compared to the PID controller.

Table 4
Impact of the error-correcting term: Values for MAE and steady-state errors for the INCAD and INCADEC controllers.

Controller	MAE	MaxASSE	MASSE
INCAD ($r = 4.5$)	0.116	0.002	0.0005
INCADEC ($r = 5.5$)	0.124	0	0

4.5. Process uncertainty

To test the robustness of the proposed inverse neural controller regarding uncertainties on process dynamics, we performed two additional tests. The first test involved a simulation where the value of the heat transfer coefficient UA was abruptly changed from 20,000 J/s K to 10,000 J/s K, assuming that initially the system was stabilized at the unstable steady-state. The four previously designed controllers, namely the INC, INCAD, INCADEC and PID controllers were asked to keep the concentration of $Na_2S_2O_3$ inside the reactor to the initial steady-state value. The dynamic responses of the controlled and the manipulated variable and the performance

Table 5
Unmeasured disturbance rejection problem: Values for Mean Absolute Error (MAE) for the four controllers.

Controller	MAE
INCADEC ($r = 10$)	0.0005
INCAD ($r = 10$)	0.0575
INC	0.2916
PID	0.0034

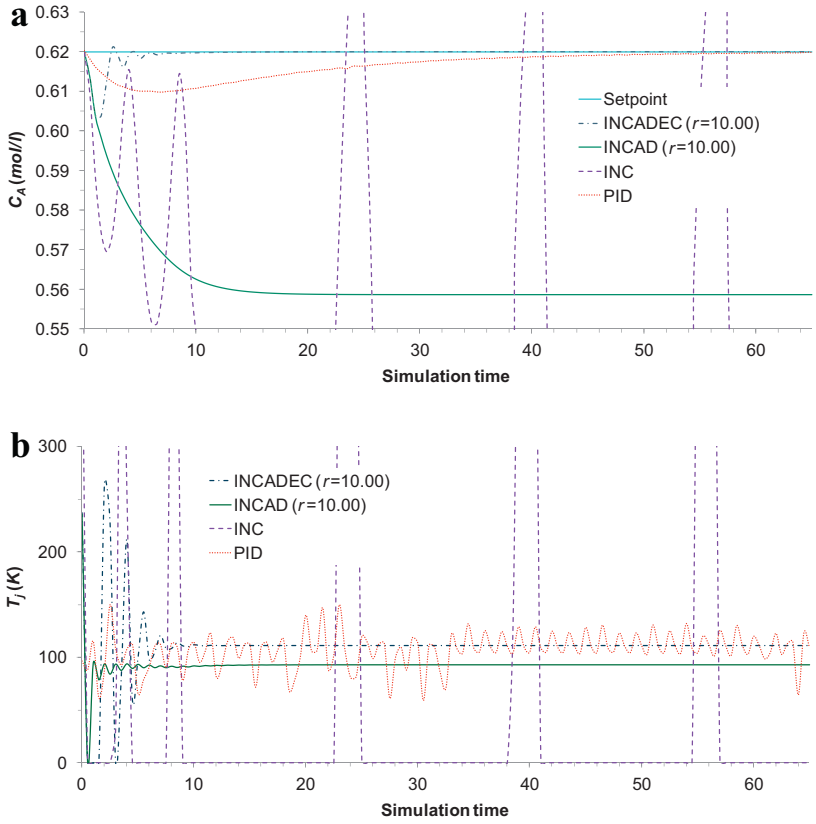

Fig. 9. Unmeasured disturbance rejection problem: (a) responses for the four controllers and (b) control actions for the four controllers.

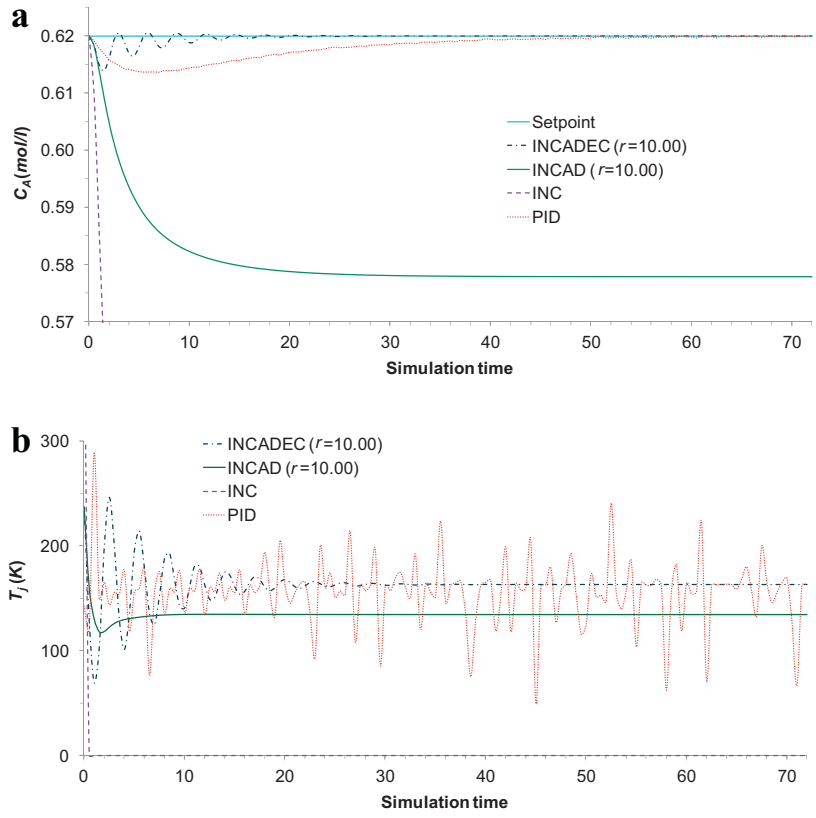

Fig. 10. Process uncertainty problem with abrupt decrease of *UA*: (a) responses for the four controllers and (b) control actions for the four controllers.

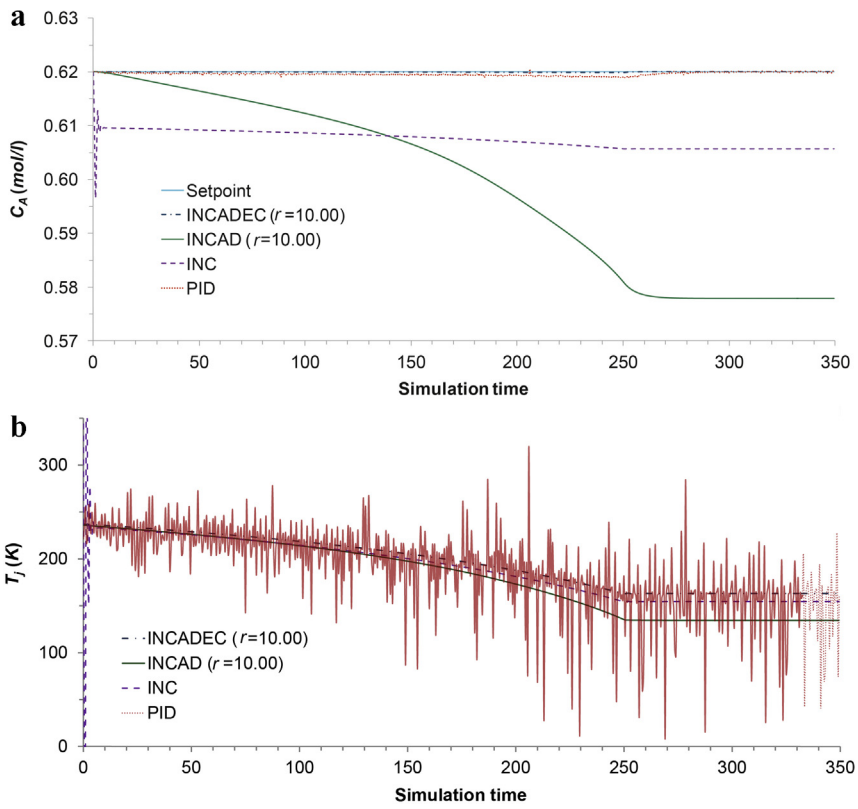

Fig. 11. Process uncertainty problem with gradual decrease of *UA*: (a) responses for the four controllers and (b) control actions for the four controllers.

Table 6
Process uncertainty problem with abrupt decrease of UA: Values for Mean Absolute Error (MAE) for the four controllers.

Controller	MAE
INCADEC (r = 10)	0.0003
INCAD (r = 10)	0.0394
INC	0.4259
PID	0.0018

Table 7
Process uncertainty problem with gradual decrease of UA: Values for Mean Absolute Error (MAE) for the four controllers.

Controller	MAE
INCADEC (r = 10)	0.00005
INCAD (r = 10)	0.02142
INC	0.01257
PID	0.00040

metrics are shown in Fig. 10a and b and Table 6, respectively. Fig. 11 a and b and Table 7, depict the respective results on the second test, where the heat transfer coefficient *UA* was gradually decreased from 20,000 J/s K to 10,000 J/s K with a rate of 40 J/s K per simulation time step, while after reaching the value of 10,000 J/s K at time step 250, the heat transfer coefficient remained constant for 100 more time periods. The results are similar to the ones described in the disturbance rejection section, i.e. the INC and INCAD controllers failed to drive the system back to the desired steady-state, while both the INCADEC and PID controllers produced zero offset. However, once more the INCADEC was clearly superior to the PID controller in terms of performance characteristics.

General remark: The proposed neural controller was trained using only collected input–output data and does not assume the existence of a first-principles model or any a priori knowledge about the process. Still, it manages to control efficiently the CSTR system through the whole operating region, including the unstable steady-state point, in many test cases, such as set-point tracking, unmeasured disturbances and process uncertainty. Moreover, it clearly outperforms the other direct inverse neural controller formulations that were tested and the PID controller. It should be noted that the PID had additional information built-in, as the procedure required for obtaining its optimal tuning parameters, assumes knowledge of the first-principles model of the process.

5. Conclusions

This work presents a novel inverse neural control scheme for nonlinear processes. The most important innovations of the proposed approach compared to existing methods are the following:

- The inverse dynamics of the process are approximated with an RBF network model trained with the FM algorithm, which presents faster computational times and increased approximation accuracy. Training is based solely on dynamic input–output data which are easy to collect by exciting the process with a random input signal.
- The method makes use of the popular in chemometrics AD concept to ensure that the controller is not used in extrapolation mode. This also provides the user with a tuning parameter, determining how conservative or aggressive the control movements will be. Additionally, in cases of major set-point or external disturbance changes, the movement from current state to the

desired set-point is broken into several smaller steps which are feasible to implement.
- At each discrete time instant, a correction term is introduced to account for model-plant mismatches and external disturbances. The error correction term follows a historical pattern, which causes adjustments of the control input until there is no discrepancy between the plant output and the desired steady-state, thus achieving offset-free control.
- The method can be considered as an alternative to explicit nonlinear MPC, as it explicitly computes a suboptimal control law based on the current state and the future set-point. However, it exhibits the important advantage of fast computation requiring only the evaluation of a nonlinear function, in contrast to the computationally intensive identification of a particular region in the input space, dictated by explicit MPC.

The proposed controller was tested on a simulated CSTR, exhibiting multiple steady-state values, including an unstable equilibrium point. The performance of the proposed methodology was compared to an inverse neural controller and a PID. The results have shown that the proposed control scheme clearly outperforms its rivals in all tested cases, including set-point tracking, unmeasured disturbance rejection and model uncertainty.

Future research plans include addressing stability issues and extending the proposed approach in order to design an inverse neural controller suitable for multiple input – multiple output (MIMO) systems.

References

[1] R. Amin, X. Yao, M. Mohammadian, R.A. Sarker, Computational intelligence in control, Idea Group Publishing (2002).
[2] H. Hong-Gui, Q. Jun-Fei, Prediction of activated sludge bulking based on a self-organizing RBF neural network, Journal of Process Control 22 (2012) 1103–1112.
[3] M. Wu, C. Xu, J. She, W. Cao, Neural-network-based integrated model for predicting burn-through point in lead–zinc sintering process, Journal of Process Control 22 (2012) 925–934.
[4] K.S. Narendra, P. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Transactions on Neural Networks 1 (1990) 4–27.
[5] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop, Neural networks for control systems—a survey, Automatica 6 (1992) 1083–1112.
[6] M.A. Hussain, Review of the applications of neural networks in chemical process control-simulation and online implementation, Artificial Intelligence in Engineering 13 (1999) 55–68.
[7] M. Mohammadzaheri, L. Chen, S. Grainger, A critical review of the most popular types of neural control, Asian Journal of Control 14 (2012) 1–11.
[8] B.M. Akesson, H.T. Toivonen, A neural network model predictive controller, Journal of Process Control 16 (2006) 937–946.
[9] M. Lawrynczuk, P. Tatjewski, Nonlinear predictive control based on neural multi-models, International Journal of Applied Mathematics and Computation 20 (2010) 7–21.
[10] N.T. Tai, K.K. Ahn, A hysteresis functional link artificial neural network for identification and model predictive control of SMA actuator, Journal of Process Control 22 (2012) 766–777.
[11] E. Aggelogiannaki, H. Sarimveis, Nonlinear model predictive control for distributed parameter systems using data driven artificial neural network models, Computers and Chemical Engineering 32 (2008) 1233–1245.
[12] A. Alexandridis, H. Sarimveis, Nonlinear adaptive model predictive control based on self-correcting neural network models, AIChE Journal 51 (2005) 2495–2506.
[13] X. Ruan, M. Ding, D. Gong, J. Qiao, On-line adaptive control for inverted pendulum balancing based on feedback-error-learning, Neurocomputing 70 (2007) 770–776.
[14] S.S. Ge, C. Yang, T.H. Lee, Adaptive predictive control using neural network for a class of pure-feedback systems in discrete time, IEEE Transactions on Neural Networks 19 (2008) 1599–1614.
[15] J. Na, X. Ren, C. Shang, Y. Guo, Adaptive neural network predictive control for nonlinear pure feedback systems with input delay, Journal of Process Control 22 (2012) 194–206.
[16] J.H. Lee, Model predictive control: review of the three decades of development, International Journal of Control and Automation 9 (2011) 415–424.
[17] K. Graichen, A. Kugi, Stability and incremental improvement of suboptimal MPC without terminal constraints, International Journal of Automation and Control 55 (2010) 2576–2580.
[18] A. Bemporad, M. Morari, V. Dua, E.N. Pistikopoulos, The explicit linear quadratic regulator for constrained systems, Automatica 38 (2002) 3–20.
[19] P. Patrinos, H. Sarimveis, A new algorithm for solving convex parametric quadratic programs based on graphical derivatives of solution mappings, Automatica 46 (2010) 1405–1418.
[20] T.A. Johansen, Approximate explicit receding horizon control of constrained nonlinear systems, Automatica 40 (2004) 293–300.
[21] A. Grancharova, T.A. Johansen, Computation, approximation and stability of explicit feedback min–max nonlinear model predictive control, Automatica 45 (2009) 1134–1143.
[22] Y. Wang, S. Boyd, Fast model predictive control using on-line optimization, IEEE Transactions on Automatic Control 18 (2010) 267–278.
[23] P. Patrinos, P. Sopasakis, H. Sarimveis, A global piecewise smooth Newton method for fast large-scale model predictive control, Automatica 47 (2011) 2016–2022.
[24] M. Lawrynczuk, Efficient nonlinear predictive control of a biochemical reactor using neural models, Bioprocess and Biosystems Engineering 32 (2009) 301–312.
[25] M. Lawrynczuk, On improving accuracy of computationally efficient nonlinear predictive control based on neural models, Chemical Engineering Science 66 (2011) 5253–5267.
[26] K.J. Hunt, D. Sbarbaro, Neural networks for nonlinear internal model control, IEE Proceedings D 138 (1991) 431–438.
[27] C. Kambhampati, R.J. Craddock, M. Tham, K. Warwick, Inverse model control using recurrent networks, Mathematics and Computers in Simulation 51 (2000) 181–199.
[28] G.L. Plett, Adaptive inverse control of linear and nonlinear systems using dynamic neural networks, IEEE Transactions on Neural Networks 14 (2003) 360–376.
[29] D.M. Psichogios, L.H. Ungar, Direct and indirect model based control using artificial neural-networks, Industrial and Engineering Chemistry Research 30 (1991) 2564–2573.
[30] E.P. Nahas, M.A. Henson, D.E. Seborg, Nonlinear internal model control strategy for neural network models, Computers and Chemical Engineering 16 (1992) 1039–1057.
[31] W. Daosud, P. Thitiyasook, A. Arpornwichanop, P. Kittisupakorn, M.A. Hussain, Neural network inverse model-based controller for the control of a steel pickling process, Computers and Chemical Engineering 29 (2005) 2110–2119.
[32] K. Vukadinovic, D. Teodorovic, G. Pavkovic, A neural network approach to the vessel dispatching problem, European Journal of Operational Research 102 (1997) 473–487.
[33] K.V. Zmeu, B.S. Notkin, P.A. Dyachenko, V.A. Kovalev, Fast predictive inverse neurocontrol: comparative simulation and experiment, in: WCCI 2012 IEEE World Congress on Computational Intelligence, Brisbane, Australia, 2012.
[34] M.A. Hussain, P. Kittisupakorn, W. Daosud, Implementation of neural-network-based inverse-model control strategies on an exothermic reactor, Scienceasia 27 (2001) 41–50.
[35] E. Hossaini-asl, M. Shahbazian, Nonlinear dynamic system control using wavelet neural network based on sampling theory, in: IEEE International Conference on Systems, Man and Cybernetics (SMC), San Antonio, TX, USA, 2009.
[36] D.B. Anuradha, G.P. Reddy, J.S.N. Murthy, Direct Inverse neural network control of a continuous stirred tank reactor (CSTR), in: International MultiConference of Engineers and Computer Scientists, 2009 Vol II (IMECS), Hong Kong, 2009.
[37] Y. Ge, S. Ma, X. Luo, Direct inverse model control based on a new improved cmac neural network, Lecture Notes in Computer Science 6215 (2010) 25–32.
[38] M.A. Hussain, L.S. Kershenbaum, Implementation of inversemodel-based control strategy using neural networks on a partially simulated exothermic reactor, Transactions of the Institution of Chemical Engineers 78 (2000) 299.
[39] M.A. Denai, F. Palis, A. Zeghbib, Modeling and control of non-linear systems using soft computing techniques, Applied Soft Computing 7 (2007) 728–738.
[40] A. Alexandridis, H. Sarimveis, G. Bafas, A new algorithm for online structure and parameter adaptation of RBF networks, Neural Network 16 (2003) 1003–1017.
[41] J. Moody, C. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation 2 (1989) 281–294.
[42] H. Sarimveis, A. Alexandridis, G. Tsekouras, G. Bafas, A fast and efficient algorithm for training radial basis function neural networks based on a fuzzy partition of the input space, Industrial and Engineering Chemistry Research 41 (2002) 751–759.
[43] F. Sahigara, K. Mansouri, D. Ballabio, A. Mauri, V. Consonni, R. Todeschini, Comparison of different approaches to define the applicability domain of QSAR models, Molecules 17 (2012) 4791–4810.
[44] K. Ninos, C. Giannakakis, I. Kompogiannis, I. Stavrakas, A. Alexandridis, Nonlinear control of a DC-motor based on radial basis function neural networks, in: IEEE International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), Istanbul, Turkey, 2011.
[45] A. Alexandridis, H. Sarimveis, K. Ninos, A radial basis function network training algorithm using a non-symmetric partition of the input space—application to a model predictive control configuration, Advances in Engineering Software 42 (2011) 830–837.
[46] A. Alexandridis, H. Sarimveis, Control of processes with multiple steady states using MPC and RBF neural networks, in: European Symposium on Computer-Aided Process Engineering (ESCAPE-21), Thessaloniki, Greece, 2011.
[47] A. Alexandridis, E. Chondrodima, H. Sarimveis, Radial basis function network training using a nonsymmetric partition of the input space and particle swarm optimization, IEEE Transactions on Neural Networks and Learning Systems 24 (2013) 219–230.

[48] J. Nie, Fuzzy control of multivariable nonlinear servomechanisms with explicit decoupling scheme, IEEE Transactions on Fuzzy Systems 5 (1997) 304.
[49] U. Maeder, F. Borrelli, M. Morari, Linear offset-free model predictive control, Automatica 45 (2009) 2214–2222.
[50] M. Morari, U. Maeder, Nonlinear offset-free model predictive control, Automatica 48 (2012) 2059–2067.
[51] N. Kazantzis, C. Kravaris, Synthesis of state feedback regulators for nonlinear processes, Chemical Engineering Science 55 (2000) 3437–3449.
[52] D.E. Seborg, T.F. Edgar, D.A. Mellichamp (Eds.), Process Dynamics and Control, 2 ed., John Wiley & Sons, Inc., Hoboken, NJ, 2004.