ELSEVIER

Chemical Engineering Journal 76 (2000) 1–16

Chemical Engineering Journal

www.elsevier.com/locate/cej

Control of nonisothermal CSTR with time varying parameters via dynamic neural network control (DNNC)

M. Nikravesh [a,*], A.E. Farell[1,b], T.G. Stanford [b]

[a] Earth Sciences Division of Lawrence Berkeley National Laboratory and Department of Materials Science and Mineral Engineering, University of California at Berkeley, Berkeley, CA 94720, USA

[b] Department of Chemical Engineering, University of South Carolina, Columbia, SC 29208, USA

Received 2 February 1999; accepted 16 June 1999

Abstract

Dynamic neural network control (DNNC) is a model predictive control strategy potentially applicable to nonlinear systems. It uses a neural network to model the process and its mathematical inverse to control the process. The advantages of single hidden layer DNNC are threefold: First, the neural network structure is very simple, having limited nodes in the hidden layer and output layer for the SISO case. Second, DNNC offers potential for better initialization of weights along with fewer weights and bias terms. Third, the controller design and implementation are easier than control strategies such as conventional and hybrid neural networks without loss in performance. The objective of this paper is to present the basic concept of single hidden layer DNNC and illustrate its potential. In addition, this paper provides a detailed case study in which DNNC is applied to the nonisothermal CSTR with time varying parameters including activation energy (i.e., deactivation of catalyst) and heat transfer coefficient (i.e., fouling). DNNC is compared with PID control. Although it is clear that DNNC will perform better than PID, it is useful to compare PID with DNNC to illustrate the extreme range of the nonlinearity of the process. This paper represents a preliminary effort to design a simplified neural network-based control approach for a class of nonlinear processes. Therefore, additional work is required for investigation of the effectiveness of this approach for other chemical processes such as batch reactors. The results show excellent DNNC performance in the region where conventional PID control fails. ©2000 Elsevier Science S.A. All rights reserved.

Keywords: DNNC; Nonisothermal CSTR; Time varying parameters

Nomenclature

a_i	unit step response coefficients, DMC convolution model coefficients
A	area
B1	hidden layer bias
B2	output layer bias
C_A	concentration of component A
C_p	heat capacity
$d(k)$	unmodeled factor or disturbance effects on the output
E	activation energy
f	transfer function
h	heat transfer coefficient for CSTR
h	filter transfer function (time domain), Eq. (42)
ΔH	heat of reaction
k_0	rate constant
n_u	number of input nodes correspond to manipulated input
n_y	number of input nodes correspond to controlled output
N	number of time intervals needed to describe the process dynamics and $a_i = a_N$ for $i \geq N$
NN	network transfer function (time domain)
q	feed flow rate
T	temperature
$\Delta u(k)$	change in the input (manipulated variable) defined as $u(k) - u(k-1)$
$\boldsymbol{uy}$	as defined in Eq. (40)
V	volume of the tank
$\boldsymbol{W1}$	$P \times M$ matrix, lower triangular matrix as defined in Eq. (A.4)
W2	hidden/output layer weight
$\boldsymbol{W\Theta}$	vector which include all the elements of the weights and bias terms, Eqs. (39) and (41)
x,X	input to the network (neuron)
y	model output

* Corresponding author. Present address: Energy and Geoscience Institute, University of Utah, Salt Lake City, Utah 84108.

[1] Present address: Union Camp Corp., PO Box B, Eastover, SC 29022.

y_ℓ	output from hidden layer nodes
y_m	output measurement
y^*	output due to input moves up to the present time
y^set	setpoint
Greek letters	
σ	as defined in Eq. (30)
α	coefficient of $y_\mathrm{m}(k)$, as defined in Eqs. (32) and (33)
β	as defined in Eq. (30)
γ	output weighting parameter
φ	filter constant as defined in Eqs. (47) and (48)
λ	move suppression parameter
ν	filter output
θ	bias
ρ	density of reactor contents
τ_d	time delay
E	objective function, Eq. (41)
$\Gamma 1$	$P \times P$ diagonal matrix, tuning parameter
$\Lambda 1$	$M \times M$ diagonal matrix, tuning parameter
Subscripts	
1	input layer, hidden layer
2	output layer
c	coolant
f	feed

1. Introduction

One of the difficulties in analyzing the dynamic response of industrial processes is the fact that they are nonlinear. Neural networks offer the ability to generate nonlinear models for systems which are difficult to model from first principles. Neural networks with three layers can approximate any nonlinear function and generate complex decision regions for input–output mapping [1–2]. This fact suggests that they hold great promise for modeling very complicated nonlinear systems and offer a cost-effective solution to control highly nonlinear processes. To date, the backpropagation neural network has been applied successfully in many process modeling and identification applications [3–9]. This fact suggests that neural networks, in conjunction with suitable control strategy such as model-based control [8–12], state-space and geometric control [13–15], and neuro-fuzzy control [16] can be used to control nonlinear systems.

In recent process control applications, neural network models are applied in control strategies in either direct or indirect methods. In the direct method, the neural network as a controller is trained to learn the inverse of the process dynamics. Since the process is modeled with a separate neural network, the controller is not the exact inverse of the process model and offset cannot be eliminated. In addition, this approach does not provide tuning parameters and updating the process model and controller model must be done separately. The concept employed in the indirect method is to train the neural network model of the process to predict future outputs from past and present inputs and outputs. In this approach, the inverse of the model at each sampling time must be calculated via an optimization routine to calculate controller outputs. Since the neural network model is frequently complex, calculation of its mathematical inverse is difficult and time consuming.

During the past few years, several backpropagation neural network control algorithms have been proposed. Neural networks are now widely used in many nonlinear control applications [3,4,8,11–13,16–18]. Recently, neural networks have been employed in model predictive control (MPC) [9–11], internal model control (IMC) [12], dynamic matrix control (DMC) [8,10,19,20], and Adaptive control [21,22]. The neural network models which have been proposed for process control are complex and have several nodes in the input and hidden layers, as well as a large number of weights and bias terms. These weights and bias terms are usually initialized randomly, using no prior process knowledge to reduce network training time.

In this paper, Dynamic neural network control (DNNC) is presented as a control strategy which uses a neural network to model the process and then applies the mathematical inverse of the process model as the controller. DNNC falls into the large class of MPC, many of which are now widely used in industry [10,23]. The DNNC strategy differs from previous neural network controllers, neural network DMC, and MPC hybrid controllers because it offers physical meaning to the parameters.

One of the keys to the design of a reliable control strategy employing neural network models is to understand the neural network structure and to establish the relationship between the process data and neural network parameters (weights and biases). The network structure of single hidden layer DNNC is very simple, comprising limited input and hidden layer nodes and an output layer node for the SISO case. Yet in the nonlinear cases studied thus far, DNNC performs as well as, if not better than, more complex neural network control strategies. DNNC offers potential for fewer weights and bias terms, therefore, better initialization of weights and fast on-line updating of weights are possible [18]. In addition, DNNC provides tuning parameters which are analogous to those of DMC.

Nonlinear, time varying behavior is common in chemical processes. When a change in the process parameters occurs, the controller frequently needs to be retuned in order to maintain satisfactory performance. Retuning the controller is usually time consuming requiring a combination of operational experience and trial-and-error. We thus explore the usefulness of DNNC for control of such systems. In this paper, the performance of DNNC is tested on a process with severe nonlinearities. We apply DNNC to a nonisothermal CSTR with time varying activation energy (i.e., deactivation of catalyst) and heat transfer coefficient (i.e., fouling). The CSTR was chosen for this case study because the dynamic behavior of the CSTR has been studied extensively and it is well known to exhibit strong parametric sensivity [12,14,18,24,25]. More importantly, the CSTR model has become one of the standard test applications for theoretical results in the area of nonlinear control [12,14,24–26].

The objective of this paper is to present the basic concept of DNNC. The structure of the paper will be as follows. First, a brief overview of neural networks trained with the back-propagation algorithm will be given. Next, the DNNC model will be developed and the relationship between DNNC and DMC will be presented. Then, DNNC will be applied to an exothermic CSTR with constant parameters and its performance will be compared to PID control strategies. Finally, DNNC will be applied to the CSTR with time varying parametric behavior. Its performance will be demonstrated via computer simulation and will be compared to traditional PID control.

2. The backpropagation neural network

Details of the backpropagation neural network are available in the literature [1,2]. Therefore, only the important characteristics of the network will be mentioned here. The typical backpropagation neural network has an input layer, an output layer, and at least one hidden layer. Each layer is fully connected to the succeeding layer with corresponding weights. In a neural network, the nonlinear elements are called nodes, neurons, or processing elements.

Consider a single neuron with a transfer function ($y_1^{(i)} = f(z^{(i)})$), connection weights, w_j, and node threshold, θ. For each pattern I,

$$Z^{(i)} = X_1^{(i)}W_1 + X_2^{(i)}W_2 + \cdots + X_N^{(i)}W_N + \theta \quad \text{for } i = 1, \ldots, P. \tag{1}$$

All patterns may be represented in matrix notation as,

$$\begin{bmatrix} z^{(1)} \\ z^{(2)} \\ \vdots \\ z^{(P)} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_N^{(1)} & 1 \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_N^{(2)} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{(P)} & x_2^{(P)} & \cdots & x_N^{(P)} & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \\ \theta \end{bmatrix} \tag{2}$$

and

$$\boldsymbol{y}_1 = F(\boldsymbol{z}). \tag{3}$$

In more compact notation,

$$\boldsymbol{z} = \boldsymbol{X}_1\boldsymbol{w}_\theta = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\theta}, \tag{4}$$

where

$$\boldsymbol{w}_\theta = [\boldsymbol{w}^{\mathrm{T}}|\theta]^{\mathrm{T}}, \tag{5}$$

$$\boldsymbol{X}_1 = [\boldsymbol{X}|\mathbf{1}] \tag{6}$$

and

$\boldsymbol{l}$	column vector of ones with P rows,
$\boldsymbol{X}$	$P \times N$ matrix with N input and P patterns,
$\boldsymbol{\theta}$	bias vector, vector with P rows of θ,
$\boldsymbol{w}$	weights, vector with N rows.

During learning, the information is propagated back through the network and used to update the connection weights. The objective function for the training algorithm is usually set up as a squared error sum,

$$E = \frac{1}{2}\sum_{i=1}^{P}(y_{(\text{observed})}^{(i)} - y_{(\text{prediction})}^{(i)})^2. \tag{7}$$

This objective function defines the error for the observed value at the output layer which is propagated back through the network. During training, the weights are adjusted to minimize this sum of squared errors.

3. Dynamic neural network control strategy (DNNC), basic concept

This section develops the DNNC algorithm. Because DNNC is analogous to DMC, a brief explanation of the basic DMC algorithm is provided so that the similarities between DMC and DNNC can be established and so that notation is clear. Then the equations for single hidden layer DNNC will be developed. Finally the training procedure for DNNC will be presented.

3.1. DMC model prediction: (linear input–output model)

The DMC algorithm is well-established and discussed throughout the literature [26–28]. Consider a linear dynamic single-input/single-output system. A discrete representation of the process dynamics with a step-response model is given by

$$y(k+j) = \sum_{i=1}^{j} a_i\, \Delta u(k-i+j) + y^*(k+j) + d(k+j), \tag{8}$$

where

$$y^*(k+j) = y_0(k+j) + \sum_{i=j+1}^{N} a_i \Delta u(k+j-i), \tag{9}$$

$$y_0(k+j) = a_N u(k-N+j-1), \tag{10}$$

$$d(k) = y_m(k) - y(k). \tag{11}$$

and

k	discrete time,
$y(k)$	model output,
$\Delta u(k)$	change in the input (manipulated variable) defined as $u(k) - u(k-1)$,
$d(k)$	unmodeled disturbance effects on the output,
a_i	unit step response coefficients,
N	number of time intervals needed to describe the process dynamics (note: $a_i = a_N$ for $i \geq N$),

$y_m(k)$	current feedback measurement,
$y^*(k+j)$	predicted output at $k+j$ due to input moves up to k.

In the absence of any additional information, it is assumed that

$$d(k+j) = d(k). \tag{12}$$

For N manipulated variables considered during the time interval of $k-N$ to $k+N$, the equation for DMC may be represented as,

$$\begin{bmatrix} \Delta y(k+1) \\ \Delta y(k+2) \\ \Delta y(k+3) \\ \vdots \\ \Delta y(k+N) \\ \vdots \\ \Delta y(K+P) \end{bmatrix} = \begin{bmatrix} \Delta u(k) & \Delta u(k-1) & \Delta u(k-2) & \cdots & \Delta u(k-N+1) & \cdots & 0 & 1 \\ \Delta u(k+1) & \Delta u(k) & \Delta u(k-1) & \cdots & \Delta u(k-N+2) & \cdots & 0 & 1 \\ \Delta u(k+2) & \Delta u(k+1) & \Delta u(k) & \cdots & \Delta u(k-N+3) & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \Delta u(k+N-1) & \Delta u(k+N-2) & \Delta u(k+N-3) & \cdots & \Delta u(k) & \cdots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \Delta u(k+P-N) & \cdots & \Delta u(k-1) & 1 \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_N \\ \vdots \\ a_{P+1} \\ d(k) \end{bmatrix}, \tag{13}$$

where

$$\Delta y(k+j) = y(k+j) - y_0(k+j).$$

In matrix notation,

$$\Delta \boldsymbol{y} = \Delta \boldsymbol{U_1} \boldsymbol{a}_\mathrm{d} \tag{14}$$

Comparing Eq. (14) for DMC with Eq. (4) for the neural network, we see that the following analogy may be drawn:

$$\Delta \boldsymbol{y} = \boldsymbol{z},\ \Delta \boldsymbol{U}_1 = \boldsymbol{X}_1,\ \boldsymbol{w}_\theta = \boldsymbol{a}_\mathrm{d}. \tag{15}$$

3.2. DMC controller design

The DMC model equation is defined by Eq. (8). By replacing y with the desired value, y^{set} and rearranging, the DMC equation becomes,

$$\begin{bmatrix} y^{\mathrm{set}} - y^*(k+1) - d(k) \\ \vdots \\ y^{\mathrm{set}} - y^*(k+P) - d(k) \end{bmatrix} = \boldsymbol{e}\,(k+1) = \boldsymbol{A}\ \Delta \boldsymbol{u}\ (k), \tag{16}$$

where

$$\boldsymbol{A} = \begin{bmatrix} a_1 & 0 & 0 & \cdots & 0 \\ a_2 & a_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_N & a_{N-1} & a_{N-2} & \cdots & a_{N-M+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_P & a_{P-1} & a_{P-2} & \cdots & a_{P-M+1} \end{bmatrix}, \tag{17}$$

and $\boldsymbol{e}(k+1)$ is a P dimensional vector of predicted deviations from the setpoint.

The following objective function is used to find the M future controller moves $\Delta u(k), \ldots, \Delta u(k+M-1)$,

$$\underset{\Delta u}{\mathrm{Min}} \left\{ \left[\sum_{i=1}^{P} \gamma^2 [y^{\mathrm{set}}(k+i) - y(k+i)]^2 \right] + \left[\sum_{j=1}^{M} \lambda^2 [\Delta u(k+M-j)]^2 \right] \right\}. \tag{18}$$

The solution of such least-squares problems is given by,

$$\Delta \boldsymbol{u}(k) = \left[\boldsymbol{A}^\mathrm{T} \boldsymbol{\Gamma}^\mathrm{T} \boldsymbol{\Gamma}\, \boldsymbol{A} + \boldsymbol{\Lambda}^\mathrm{T} \boldsymbol{\Lambda} \right]^{-1} \times \boldsymbol{A}^\mathrm{T} \boldsymbol{\Gamma}^\mathrm{T} \boldsymbol{\Gamma} \quad [\boldsymbol{e}\,(k+1)], \tag{19}$$

where

$$\boldsymbol{\Lambda} = \mathrm{diag} \Big(\underbrace{\lambda\ \lambda\ \cdots\ \lambda}_{|\leftarrow M \rightarrow|} \Big) \tag{20}$$

λ = move suppression parameter,

$$\boldsymbol{\Gamma} = \mathrm{diag} \Big(\underbrace{\gamma\ \gamma\ \cdots\ \gamma}_{|\leftarrow P \rightarrow|} \Big) \tag{21}$$

γ = output weighting parameter.

By analogy, if the neural network is comprised of only a single neuron with a linear transfer function, then the controller would be given by

$$\boldsymbol{x} = \left[\left[\boldsymbol{W}^{\mathrm{T}}\boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{\Gamma}\,\boldsymbol{W} + \boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\Lambda}\right]^{-1}\boldsymbol{W}^{\mathrm{T}} \quad \boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{\Gamma} \quad [z]\right], \tag{22}$$

where

$$\boldsymbol{W} = \begin{bmatrix} w_1 & 0 & 0 & \cdots & 0 \\ w_2 & w_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_N & w_{N-1} & w_{N-2} & \cdots & w_{N-M+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_P & w_{P-1} & w_{P-2} & \cdots & w_{P-M+1} \end{bmatrix} \tag{23}$$

$$\boldsymbol{x} \equiv \Delta\boldsymbol{u}, \ \boldsymbol{z} \equiv \boldsymbol{e}\,(k+1). \tag{24}$$

3.3. The DNNC process model

For a single future move in Eq. (8) we have,

$$\Delta\boldsymbol{u} = [\Delta u(k) \quad 0 \quad \ldots \quad 0]^{\mathrm{T}}.$$

Therefore,

$$y(k+1) = y^{*}(k+1) + a_1\Delta u(k) + d(k+1). \tag{25}$$

For the nonlinear case, a_1 is a function of $\Delta u(k)$ and the system state, therefore, we can generalize Eq. (25) as follows,

$$y(k+1) = y^{*}(k+1) + a_1(k)\Delta u(k) + d(k+1) \tag{26}$$

or equivalently,

$$e(k+1) = a_1(k)\Delta u(k). \tag{27}$$

Solving Eq. (26) for one future move ($j=1$), and substituting for $y^{*}(k+1)$ from Eqs. (9) and (10),

$$\begin{aligned} y(k+1) = {} & a_1(k)\cdot\Delta u(k) + a_2(k)\cdot\Delta u(k-1) + a_3(k) \\ & \cdot\Delta u(k-2) + \cdots + a_N(k) \\ & \cdot\Delta u(k+1-N) + a_N(k)\cdot u(k-N) + d(k+1) \end{aligned} \tag{28}$$

with $d(k+1)$ is defined as follows:

$$d(k+1) = y_m(k+1) - y(k+1) \tag{29}$$

and in terms of $y_m(k)$ and $y(k)$,

$$d(k+1) = \sigma(k)y_m(k) - \beta(k)y(k), \tag{30}$$

where $\sigma(k)$ and $\beta(k)$ are time varying parameters. Substituting for $y(k)$ from Eq. (11) into Eq. (30) gives,

$$d(k+1) = [\sigma(k) - \beta(k)]\,y_m(k) + \beta(k)d(k). \tag{31}$$

Eq. (30) clearly shows that $d(k+1)$ is a function of $y_m(k)$, $d(k)$, and the parameters $\beta(k)$ and $\sigma(k)$. Define $\alpha(k)=\sigma(k)-\beta(k)$ so Eq. (31) may be written as

$$d(k+1) = \alpha(k)y_m(k) + \beta(k)d(k). \tag{32}$$

It is noted that by taking $\alpha(k)=0$ and $\beta(k)=1$, Eq. (32) reduces to Eq. (11) for DMC and $d(k+1)$ is dependent only on $d(k)$.

Substituting Eq. (32) into Eq. (28) and simplifying one obtains,

$$\begin{aligned} y(k+1) = {} & a_1(k)\cdot\Delta u(k) + a_2(k)\cdot\Delta u(k-1) + a_3(k) \\ & \cdot\Delta u(k-2) + \cdots + a_{N-1}(k)\Delta u(k-N+2) \\ & + a_N(k)\cdot u(k-N+1) + \alpha(k)y_m(k) + \beta(k)d(k) \\ = {} & \boldsymbol{a}(k)^{\mathrm{T}}\Delta\boldsymbol{uy}(k) + \beta(k)d(k), \end{aligned} \tag{33}$$

where

$$\begin{aligned} \Delta\boldsymbol{uy}(k) = {} & [\Delta u(k) \quad \Delta u(k-1) \quad \cdots \quad \Delta u(k-N+2) \\ & \times u(k-N+1) \quad y_m(k)]^{\mathrm{T}}, \quad \boldsymbol{a}(k) = [a_1(k) \\ & \times a_2(k) \quad \ldots \quad a_{N-1}(k) \quad a_N(k) \\ & \times a_{N+1}(k)]^{\mathrm{T}} \quad a_{N+1}(k) = \alpha(k). \end{aligned} \tag{34}$$

Eq. (33) shows that $y(k+1)$ is a function of the independent variables $\Delta\boldsymbol{uy}(k)$ and the time varying parameters $a(k)$. (We will assume that in the training data $d(k)=0$.) In a more general form, $y(k+1)$ is given by,

$$y(k+1) = g(\Delta\boldsymbol{uy}(k), \boldsymbol{a}(k)). \tag{35}$$

In this study, we use a neural network model for nonlinear input/output mapping given by Eq. (35) with the input structure of the network as defined by Eq. (33). In general, for nonlinear processes the single hidden layer DNNC process model is defined by,

$$\begin{aligned} & y(k+1) = w_2 f(\mathbf{w}_1^{\mathrm{T}}\Delta\boldsymbol{uy} + B_1) + B_2, \\ & \Delta\boldsymbol{uy} = [\Delta u(k) \quad \Delta u(k-1) \quad \cdots \quad \Delta u(k-N+2) \\ & u(k-N+1) \quad y_m(k)]^{\mathrm{T}}, \quad \boldsymbol{w}_1 = [w1_1 \quad w1_2 \quad \ldots \\ & w1_{N-1} \quad w1_N \quad w1_{N+1}]^{\mathrm{T}} \end{aligned} \tag{36}$$

with the transfer function f typically defined by the hyperbolic tangent function,

$$f(z) = \frac{\exp(z) - \exp(-z)}{\exp(z) + \exp(-z)}. \tag{37}$$

It is important to note that there is no restriction on the number of hidden layers or the transfer function in the output and hidden layers.

3.3.1. Training the DNNC model

With the initial guess for the network weights and biases,

$$\begin{aligned} & \boldsymbol{w}_1 = [\boldsymbol{a}^{\mathrm{T}}|\alpha]^{\mathrm{T}}, \quad w_2 = 1, \quad B_1 = B_2 = 0, \\ & w_{1,(N+1)} = \alpha = \varepsilon, \end{aligned} \tag{38}$$

where

$$\varepsilon \cong 0.0, \ (\text{e.g. } \varepsilon = 0.001).$$

The DNNC model is trained with input/output data using the backpropagation algorithm. In applications, the weights and bias terms would be updated either using the back-propagation algorithm or a recursive algorithm if necessary [18,26]. It has been shown [18,26] that DNNC offers potential for fewer weights and bias terms, therefore, better initialization of weights and fast on-line updating of weights are possible [18].

3.4. DNNC controller design

Before discussing the DNNC controller, we present a brief summary of current work with neural network controllers. This discussion will help to illustrate the advantages of DNNC. Neural networks can be used as a controller in either direct or indirect methods as discussed in the introduction. In the direct approach, the controller will be a neural network which represents the inverse of the process model, while in the indirect approach the controller algorithm inverts the neural network process model. In the direct approach, since the controller model is trained off line, the error in the controller prediction significantly affects the controller performance. This problem was addressed by Psichogios and Ungar [9]. In the indirect approach, the inverse of the process model at each sampling time must be calculated. Since the neural network model of the process is a nonlinear mapping between input and output, and the model is frequently complex, mathematical inversion of the process model is difficult. Therefore, the most popular method applies an optimization routine to find the controller output. To illustrate the complexity, we will outline the optimization approach.

The neural network process model for input/output mapping is defined as,

$$y(k+1) = \mathrm{NN}(\boldsymbol{uy}(k),\ \boldsymbol{W\Theta}(k)), \tag{39}$$

where $\boldsymbol{uy}(k)$ includes all the inputs to the network model at time (k);

$$\boldsymbol{uy}(k)^{\mathrm{T}} = [y(k), y(k-1), ..., y(k-n_y+1), u(k), u(k-1), \\ \dots, u(k-n_u+1)], \tag{40}$$

and $\boldsymbol{W\Theta}(k)$ includes all the weights and bias terms. The objective function (OF) for controller design is given by,

$$\mathrm{OF} = E(k)^2 = [E(\nu(k), \boldsymbol{uy}(k), \boldsymbol{W\Theta}(k))]^2 \\ = [\upsilon(k) - \mathrm{NN}(\boldsymbol{uy}(k), \boldsymbol{W\Theta}(k))]^2\,, \tag{41}$$

where $\nu(k)$ is a filtered output given by,

$$\nu(k) = h(d(k), y^{\mathrm{set}}(k), \nu(k-1)). \tag{42}$$

In the optimization routine, $u(k)$ is calculated to minimize the objective function defined in Eq. (41). The back-propagation algorithm may be used, in which case the error back-propagates through the network to adjust the $u(k)$ instead of adjusting the weights and bias terms as is done during training. Other methods such as quadratic programming also may be applied. Solving the optimization problem for conventional neural network models, which are complex and consist of several nodes and weights, is difficult and time consuming. In addition, there is approximation error in calculating $u(k)$. Often, this error in $u(k)$ significantly affects the controller performance. Psichogios and Ungar [9] suggest "detuning" the controller for robust performance.

Another approach to find the controller output based on the process model employs Newton's method. Several authors have employed this technique [9,12]. The following equations are used in this method at each iteration j,

$$u(k)^{[j]} = u(k)^{[j-1]} - \frac{E(k)^{[j-1]}}{\partial E(k)^{[j-1]}/\partial u(k)^{[j-1]}} \tag{43}$$

with the initial guess for $u(k)$ as follows:

$$u(k)^{[0]} = u(k-1). \tag{44}$$

Details of the calculation of the Jacobian of $E(k)$; $\left[\partial E(k)^{[j-1]}/\partial u(k)^{[j-1]}\right]$ have been presented by Nikravesh et al. [18]. Once again there is approximation error in $u(k)$ and "detuning" is required. In addition, solving this nonlinear problem by Newton's method is not trivial. Psichogios and Ungar [9] considered some problems associated with convergence of Newton's method. In their work, consideration was given to problems including poor initial guesses, pathological functions, and singular problems with vanishing derivatives.

Since the DNNC model is very simple, its inversion does not present the difficulties encountered with other methods. The inverse of the process model is exact so, convergence and offset do not present problems. In this paper, we present the DNNC controller in an IMC framework (one step ahead prediction). The neural network IMC models are restricted to systems with stable open-loop response. However, DNNC can be employed in a more general model predictive control (MPC) framework (multi-step prediction). For example, DNNC can be employed in the DMC framework. Details of such a DNNC controller are given by Appendix A.

Fig. 1 shows the block diagram for the IMC framework. The IMC version of the DNNC controller model will be designed using the following equations:

$$\Delta u(k) = \frac{\left[f^{-1}\left(\frac{\nu(k)-B_2}{w_2}\right) - B_1 - (\boldsymbol{w}_1^N)^{\mathrm{T}}\Delta\boldsymbol{uy}_1^N\right]}{w1_1}, \\ \boldsymbol{w}_1^N = [w1_2 \quad w1_3 \quad ... \quad w1_N \quad w1_{N+1}]^{\mathrm{T}}, \Delta\boldsymbol{uy}_1^N \\ = [\Delta u(k-1) \quad \Delta u(k-2) \quad ... \quad \Delta u(k-N+2) \\ \times \Delta u(k-N+1) \quad y_m(k)], \tag{45}$$

where $\nu(k)$ is a filtered output for $y(k+1)$. For the controller design, $y(k+1) = y^{\mathrm{set}}(k+1)$. The transfer function f^{-1} is the inverse of Eq. (37),

$$f^{-1}(z) = -0.5\ln\left[\frac{1-(z)}{1+(z)}\right]. \tag{46}$$

The filter may be chosen to be a pulse transfer function in

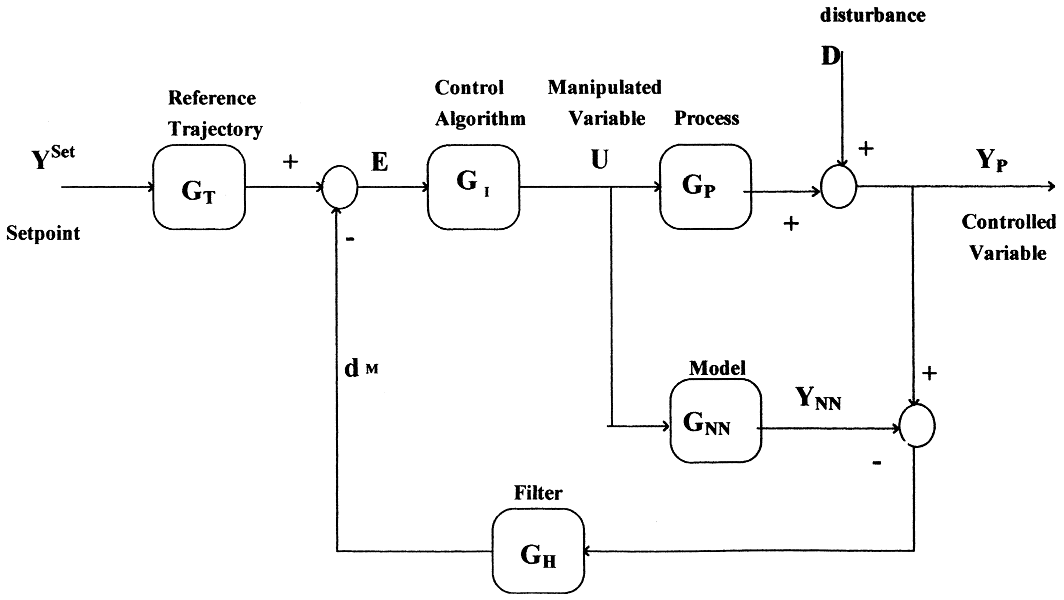

Fig. 1. Block diagram of simplified neural network IMC structure.

the form of a first-order filter. The filter equation is given by,

$$v(k) = v(k-1) + (1-\varphi)\left[y^{\text{set}}(k) - d(k) - y(k)\right]. \quad (47)$$

In training the network via the backpropagation algorithm, an appropriate scaling of inputs and $y(k+1)$ will be used so that, $-1 < y(k+1) < +1$. In this case, $(y(k+1) - B_2)/w_2$ is always bounded between -1 and 1. It is also possible to scale the inputs and $y(k+1)$ such that, $-1 < (v(k) - B_2)/w_2 < +1$, is guaranteed. Therefore, the existence of $f^{-1}((v(k) - B_2)/w_2)$ is guaranteed.

For time delay compensation, the following equation may be used,

$$\begin{aligned} v(k) = {} & \varphi v(k-1) + (1-\varphi)\left[y^{\text{set}}(k) - d(k) - y(k)\right. \\ & \left. + v(k-\tau_{\text{d}}-1)\right]. \end{aligned} \quad (48)$$

The value for φ is bounded between zero and one. Details of filter design are available throughout the literature [12,29–31]. In case of process/model mismatch, the filter may be used to ensure stability and robustness, but it can also compensate for certain types of disturbances. Garcia and Morari [32] and Deshpande [33,34] provide detailed guidelines for designing IMC filters.

3.5. *Stability analysis of neural network-based control systems*

The stability analysis of neural-based control systems is an important issue which must be considered for the design of a good neural-based control system [26,35]. There are several methods which has been proposed to study the open-loop and closed-loop stability of processes and to analyze and design control systems [36]. State-space methods are best suited for analysis and synthesis of nonlinear systems and they can be applied to the design of optimal control systems [36]. Once the systems are transformed into state-space models, nonlinear model approaches such as geometric control [13–15], neuro-fuzzy control [16,17], fuzzy logic control [37], and model-based control [8–13,19–22] can be used to design and analysis the controller performance. In addition, the Liapunov theory [36,38] can be used for stability analysis [4,26,35,39,40]. Liapunov stability theory plays an important role in the stability analysis of control systems described by state-space equations. The second method of Liapunov[35,36,38] is most commonly used and is applicable to both linear and nonlinear systems. This method is also suited for the stability of nonlinear systems for which exact solutions may be unobtainable such as neural network models. Although the second method of Liapunov is applicable to a wide class of nonlinear systems including neural network systems for stability analysis, obtaining successful results is not trivial [36]. Therefore, experience may be necessary to correctly interpret the results from the stability analysis of nonlinear systems.

3.5.1. *Dynamic neural network control (DNNC); state-space representation of DNNC and stability analysis*

The DNNC can be written in the following discrete state-space form,

$$\begin{aligned} &\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k)) + g(\boldsymbol{x}(k), u(k)), \\ &y(k) = h\left(z^{-1}(\boldsymbol{x}(k), u(k))\right), \\ &y(k+1) = h(\boldsymbol{x}(k), u(k)) \end{aligned} \quad (49)$$

with $\boldsymbol{x}(k+1)$ is given by,

$$\boldsymbol{x}(k+1) = [u(k)\ \ x_1(k)\ \ \ldots\ \ x_{N-2}(k)\ \ h(\boldsymbol{x}(k), u(k))]^{\text{T}} \quad (50)$$

and

$$f(\boldsymbol{x}(k)) = [0 \quad x_1(k) \quad x_2(k) \; \dots \; x_{N-2}(k) \quad 0]^{\mathrm{T}},$$
$$g(\boldsymbol{x}(k), u(k)) = [u(k) \quad 0 \quad 0 \; \dots \; 0 \quad 0 \quad h(\boldsymbol{x}(k), u(k))]^{\mathrm{T}},$$
$$h(\boldsymbol{x}(k), u(k)) = y(k+1) = w_2\,\Gamma(\Delta \boldsymbol{w}_1^{\mathrm{T}} \boldsymbol{uy} + B_1) + B_2. \tag{51}$$

In this study, we are interested in the stability of the overall process. In the DNNC strategy (as for any other IMC strategy (Fig. 1)) and with an exact model for the process, the stability of both the process and controller is sufficient for overall system stability. Nikravesh et al. [41] and Hernandez and Arkun [42] provide detailed guidelines for stability of neural networks.

4. Extension of the DNNC model to the MIMO case

For a MIMO system it is much more difficult to make meaningful specifications than for a SIOS systems. However, the design of multi-variable controllers in the DNNC control strategy is straightforward. The DNNC weights represent the process model and interaction between input/output and it is possible to determine the severity of the interaction by interpreting the network weights. The following sections will demonstrate the design of MIMO systems in the DNNC framework.

4.1. Extension of the DNNC model to the MIMO case in IMC framework

Design of multi-variable controllers in the DNNC control strategy in IMC framework is straightforward. The following equations will be used for process model:

$$\boldsymbol{y}(k+1) = \boldsymbol{w}_2 F(\boldsymbol{W}_1 \boldsymbol{uy}(k) + \boldsymbol{B}_1) + \boldsymbol{B}_2, \boldsymbol{uy} = [\Delta \boldsymbol{uy}^{(1)}(k) \quad \Delta \boldsymbol{uy}^{(2)}(k) \; \dots \; \Delta \boldsymbol{u}^{(j)}(k)]^{\mathrm{T}},$$
$$\Delta \boldsymbol{uy}^{(j)} = [\Delta \boldsymbol{u}^j(k) \quad \Delta \boldsymbol{u}^j(k) \; \dots \; \boldsymbol{u}^j(k - N_i + 1) \; y_m^j(k)]^{\mathrm{T}}, \boldsymbol{y}(k+1) = [y^{(1)}(k+1) \; y^{(2)}(k+1) \; \dots \; y^{(j)}(k+1)]^{\mathrm{T}} \tag{52}$$

with,

$$\boldsymbol{W}_1 = \begin{bmatrix} \boldsymbol{w}_{1,1} & \boldsymbol{w}_{1,2} & \cdots & \boldsymbol{w}_{1,j} \\ \boldsymbol{w}_{2,1} & \boldsymbol{w}_{2,2} & \cdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{w}_{j,1} & \boldsymbol{w}_{j,2} & \cdots & \boldsymbol{w}_{j,j} \end{bmatrix},$$
$$\boldsymbol{w}_{i,j} = [w_{i,1} \quad w_{i,2} \quad \dots \quad w_{i,N_j} \quad w_{i,N_j+1}]^{\mathrm{T}}, \tag{53}$$

where, $\boldsymbol{w}_{i,j}$ represents the effect of input i on the output j. Therefore, the DNNC controller will be designed using the inverse of the DNNC process model as was presented in the previous chapters for SISO case. For no interaction between outputs, all the elements of matrix $\boldsymbol{W}_1$ which are not on the main diagonal will be equal to zero. In other words, if after training the MIMO–DNNC model, all the off diagonal elements of $\boldsymbol{W}_1$ are equal to zero or a small value compared to the elements on the main diagonal, then the controller will be designed as a multi-SISO controller model.

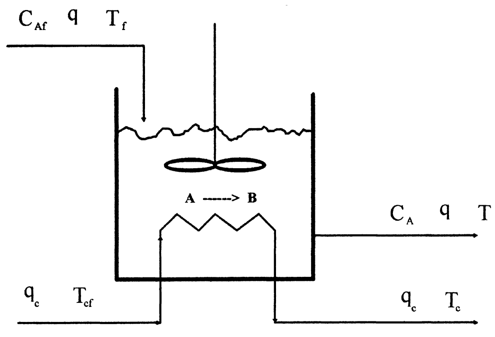

Fig. 2. Nonisothermal CSTR.

4.2. Extension of the DNNC model to the MIMO case in DMC framework

The MIMO problem formulation and equations for DNNC are the same as in the DMC case. In other words there is no difference between the form of DNNC equations and that for the DMC case, and only a change in interpretation of the variables is needed. Therefore, all properties and techniques applied to MIMO–DMC are applicable to DNNC as well. Therefore, if the DNNC controller is implemented in the DMC configuration (Appendix A as shown for SISO case), extension of SISO–DNNC to MIMO–DNNC will be straightforward and does not cause any problems (Appendix B).

5. Simulation studies

In this section, first DNNC will be applied to a nonisothermal CSTR with constant parameters and its performance will be compared to PID control. Next, DNNC will be applied to an exothermic CSTR with time varying parameters including activation energy and heat transfer coefficient. Its performance will be compared to traditional PID control. The examples illustrate improvement of DNNC over PID control.

The performance of the DNNC strategy was tested on a nonisothermal CSTR with irreversible reaction (A→B) (Fig. 2). The process model consists of two nonlinear ordinary differential equations and is given by [26]

$$\frac{\mathrm{d}C_{\mathrm{A}}}{\mathrm{d}t} = \frac{q}{V}(C_{\mathrm{Af}} - C_{\mathrm{A}}) - k_0 C_{\mathrm{A}} \exp(-\frac{E}{RT})\phi_{\mathrm{c}}(t), \tag{54}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{q}{V}(T_{\mathrm{f}} - T) + \frac{(-\Delta H) k_0 C_{\mathrm{A}}}{\rho C_p} \exp(-\frac{E}{RT})\phi_{\mathrm{c}}(t) + \frac{\rho_{\mathrm{c}} C_{\mathrm{pc}}}{\rho C_p V} q_{\mathrm{c}} \left[1 - \exp\left(-\frac{hA}{q_{\mathrm{c}} \rho C_{pc}} \phi_h(t)\right)\right] \times (T_{\mathrm{cf}} - T), \tag{55}$$

Table 1
Nominal CSTR operating condition and PID parameters

$q = 100\,\mathrm{l\,min^{-1}}$	$E/R = 9.95 \times 10^3\,\mathrm{K}$
$C_{Af} = 1\,\mathrm{mol\,l^{-1}}$	$-\Delta H = 2 \times 10^5\,\mathrm{cal\,mol^{-1}}$
$T_f = 350\,\mathrm{K}$	$\rho, \rho_c = 1000\,\mathrm{g\,l^{-1}}$
$T_{cf} = 350\,\mathrm{K}$	$C_p, C_{pc} = 1\,\mathrm{cal\,g^{-1}\,K^{-1}}$
$V = 100\,\mathrm{l}$	$q_c = 103.41\,\mathrm{l\,min^{-1}}$
$hA = 7 \times 10^5\,\mathrm{cal\,min^{-1}\,K^{-1}}$	$T = 440.2\,\mathrm{K}$
$k_0 = 7.2 \times 10^{10}\,\mathrm{min^{-1}}$	$C_A = 8.36 \times 10^{-2}\,\mathrm{mol\,l^{-1}}$
$K_c = 190.1\,\mathrm{l^2/mol^{-1}}$, $\tau_I = 0.556\,\mathrm{min^{-1}}$, $\tau_D = 0.827\,\mathrm{min}$	

where

$\phi_h(t)$	fouling coefficient,
$\phi_c(t)$	deactivation coefficient,
C_A	effluent concentration, the controlled variable,
q_c	coolant flow rate, the manipulated variable,
q	feed flow rate, disturbance,
C_{Af}	feed concentration,
T_f	feed temperatures,
T_{cf}	coolant inlet temperature.

The remaining model parameters and operating conditions are presented in Table 1.

5.1. Control of nonisothermal CSTR with constant parameters:($\phi_h(t) = 1, \phi_c(t) = 1$)

The open-loop step responses for a series of step changes in q_c is shown in Fig. 3. It is seen that the process is highly nonlinear. The DNNC process model has 16 input nodes (inputs include: current value of C_A, current and 15 previous values of q_c; scaled uniformly between 0 and 1), one node in the hidden layer with a nonlinear hyperbolic tangent transfer function, and one node in the output layer (output predictions into the future, C_A; scaled uniformly between 0 and 1). The model is trained via backpropagation algorithm with data generated with random changes in the q_c. In the DNNC control strategy, Eq. (45) is used to find the manipulated input at each sampling time. The model predicts the controlled output. We note that the DNNC structure is very simple and has small number of nodes (total number of weights and bias terms = 20).

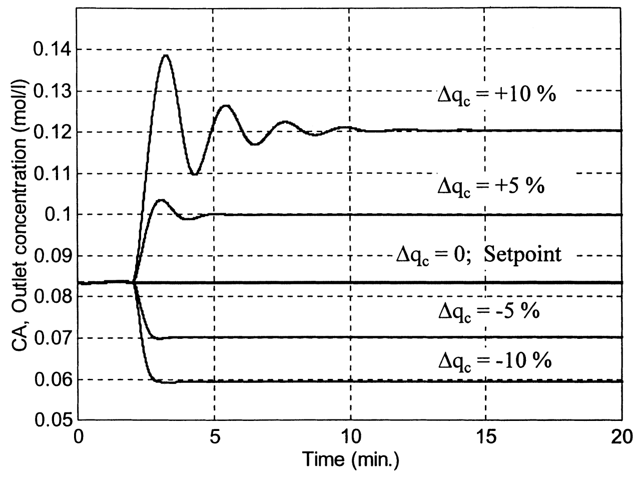

Fig. 3. Open-loop response of the CSTR for step changes in the coolant flow rate q$_c$.

To illustrate the performance of the DNNC strategy, DNNC and PID are applied to control the CSTR process. DNNC was tuned with a filter constant value of $\varphi = 0.95$. PID tuning parameters for this case study are given in Table 1. In Fig. 4, the setpoint tracking behaviors of DNNC and PID are compared. For setpoint changes, DNNC shows a faster response toward the setpoint than the PID control strategies. Fig. 5 shows the disturbance rejection performance of DNNC and PID controller strategies. For 20% change in inlet flow rate as disturbance, DNNC exhibits a faster response toward setpoint when compared to PID. In comparison to the PID, DNNC shows excellent performance with much faster response time toward the setpoint.

5.2. Control of the nonisothermal CSTR with time varying parameters

In this section, the previously developed DNNC and PID control strategies (Section 5.1) will be applied to the nonisothermal CSTR with time varying parameters. Two case studies will be examined in which the CSTR exhibits severe nonlinearities. In the first case study, the effect of fouling on the performance of DNNC and PID controller strategy will be demonstrated. In the second case study, the effect of activation energy will be presented.

5.2.1. Case study 1: time varying heat transfer coefficient

Fouling occurs when a material is deposited on a heat transfer surface during the period of process operation. In practice, it is common for heat transfer surfaces to become contaminated with deposits and this causes additional resistance to the flow of heat. There are two common behaviors in the development of a fouling film over a period of time [43]. One is the so-called asymptotic fouling. In this case, the resistance to heat transfer increases very quickly in the beginning of the operation and becomes asymptotic to a steady state value at the end. The other is the so-called linear fouling, where the fouling resistance increases linearly during the entire process operation. In this study, we assume that the fouling film develops linearly over the entire period of process operation. Therefore, the heat transfer coefficient h defined in Eq. (55) is replaced by h_d and is given by

$$h_d = \phi_h(t)h = (1 - \alpha_h t)h, \tag{56}$$

where

t	time,
h	heat transfer coefficient, cleaned,
h_d	heat transfer coefficient, scaled,
$\phi_h(t)$	fouling coefficient, $0 < \phi_h < 1$,
α_h	fouling constant.

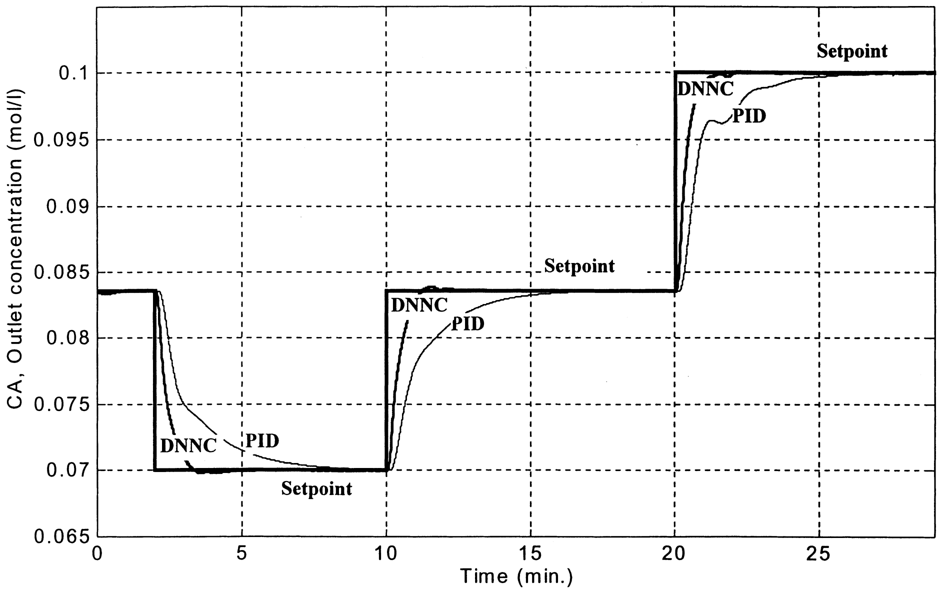

Fig. 4. Setpoint tracking performance for CSTR.

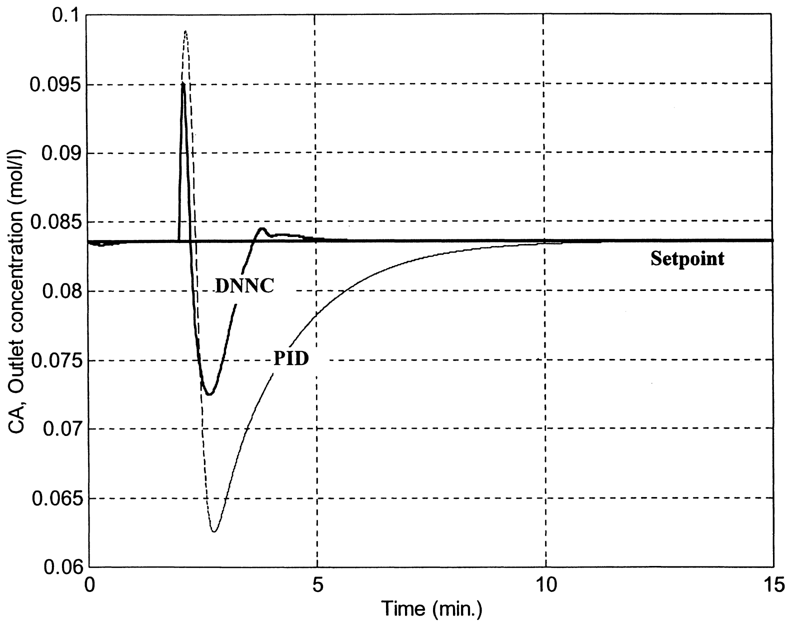

Fig. 5. Distance rejection performance for +20% change in the feed flow rate of the CSTR.

The open-loop step responses for a series of step changes in q_c is shown in Fig. 6. Fig. 7 shows the comparison between open-loop response of the CSTR with time varying heat transfer coefficient and CSTR with constant parameter for +5% step changes in the coolant flow rate, q_c. Figs. 6 and 7 show that the process does not have steady state condition and the process is nonstationary. Figs. 8 and 9 show that the setpoint tracking and disturbance rejection performance of DNNC and PID are only slightly affected by the time varying behavior of the heat transfer coefficient. DNNC exhibits faster response time toward setpoint than PID for both setpoint tracking and disturbance rejection and is able to control the process effectively.

5.2.2. Case study 2: time varying activation energy

The temperature dependence of the rate expression is usually represented by the rate constant through the Arrhenius equation;

$$k = k_0 \exp\left(-\frac{E}{RT}\right), \tag{57}$$

where

k_0	frequency factor,
E	activation energy,
R	gas constant,
T	absolute temperature.

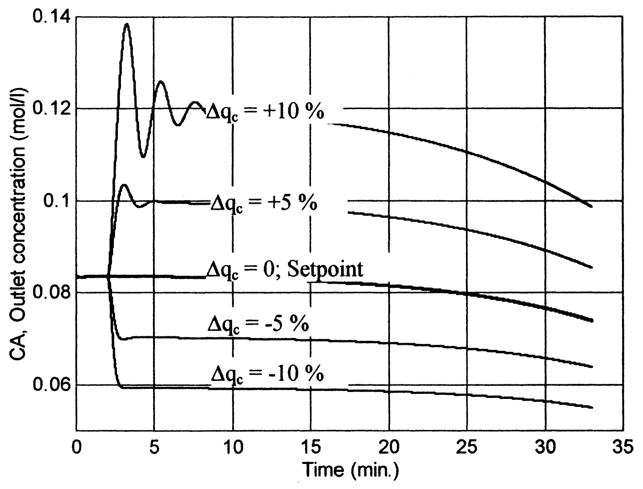

Fig. 6. Open-loop response of the CSTR with time varying heat transfer coefficient for step changes in the coolant flow rate, q_c; [Effect of fouling $\phi_h(t) = (1-\alpha_h t)$ with $\alpha_h = 0.01$].

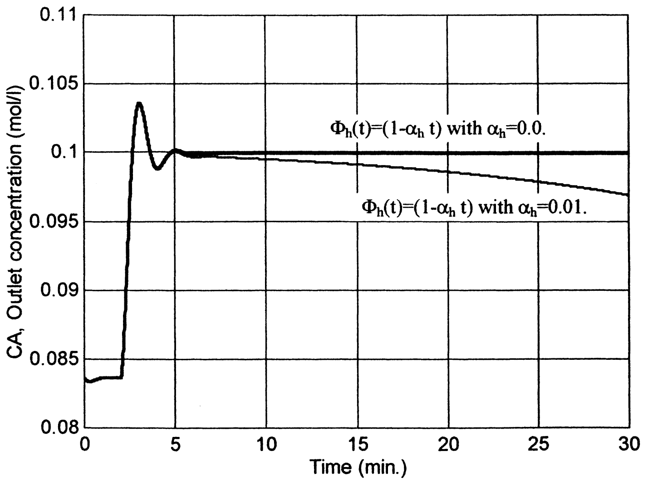

Fig. 7. Comparison between open-loop response of the CSTR with time varying heat transfer coefficient and CSTR with constant parameter for +5% step change in the coolant flow rate, q_c.

Although the activation energy is not affected by temperature in the moderate temperature range, some exceptions have been reported [44]. Other factors known to influence the activation energy include pressure (gas phase reactions) and the presence of a catalyst [43,44]. In the presence of a catalyst, an important factor which affects the rate of reaction is the deactivation of the catalyst by poisoning. Obviously, there is no exact theoretical expression for the deactivation process. However, some empirical expressions has been reported. In this case study, the general equation for k is given by,

$$k = k_0 \exp\left(-\frac{E}{RT}\right)\phi_c(t). \tag{58}$$

Table 2 shows some functions which have been proposed for ϕ_c (t) [44]. Here we will consider the first functional form

Table 2
Empirical deactivation functions

$\phi_c = 1 - \alpha t$
$\phi_c = \exp(-\alpha t)$
$\phi_c = 1/1 + \alpha t$
$\phi_c = \alpha t^{-0.5}$
$\phi_c = (1+\alpha t)^{-p}$

from Table 2 for $\phi_c(t)$ and is given by,

$$\phi_c(t) = \exp(-\alpha t). \tag{59}$$

Catalyst deactivation. The open-loop step responses for a series of step changes in q_c is shown in Figs. 10 and 11 shows the comparison between open-loop response of the CSTR catalyst deactivation and CSTR with constant parameter for +5% step changes in the coolant flow rate, q_c. Figs. 10 and 11 show that the process does not have steady state condition and the process is nonstationary. Figs. 12 and 13 show the setpoint tracking and disturbance rejection performance of DNNC and PID. Comparing Fig. 4 with Figs. 12 and 14, one can see that the performance of PID is seriously affected due to these changes. PID shows a very large offset and very slow response with extremely poor performance. While the PID control strategy fails to control the process, DNNC shows good performance. Modification of the PID strategy to get better performance is not trivial. However, fine tuning the DNNC controller is very easy and straightforward. For open-loop stable processes, a filter is introduced into the DNNC structure in order to ensure stability [26]. In addition, the stability of DNNC is improved by increasing the number of nodes at the input layer (more manipulated inputs) [26]. In the presence of model inaccuracies, the robustness of DNNC is improved by introducing a first order exponential filter. The value of φ (filter constant in Eq. (47)) is bounded between zero and one. Details of filter design are available throughout the literature [12,26,29–31]. Fig. 14 shows the setpoint tracking performance of DNNC with different filter constant φ. Increasing φ will result in the slower but smoother response. Decreasing φ will result in a faster response but with more oscillation.

Catalyst regeneration. Fig. 15 shows the open-loop step responses for a series of step changes in q_c. Fig. 15 shows that the process does not have steady state condition and the process is nonstationary. In addition, comparing Figs. 10 and 15, one can see that the behavior of the process in the positive and negative direction are quite different. Figs. 16 and 17 show the setpoint tracking and disturbance rejection performance of DNNC and PID. Comparing Fig. 4 with Figs. 16 and 17, one can see that the performance of PID is seriously affected due to these changes. PID shows unstable response with extremely poor performance and diverging from the setpoint. While the PID control strategy fails to control the process, DNNC controls the process but with large offset due to continuous change in the process parameters. Fine tuning the DNNC controller is very easy and straightforward and discussed earlier.

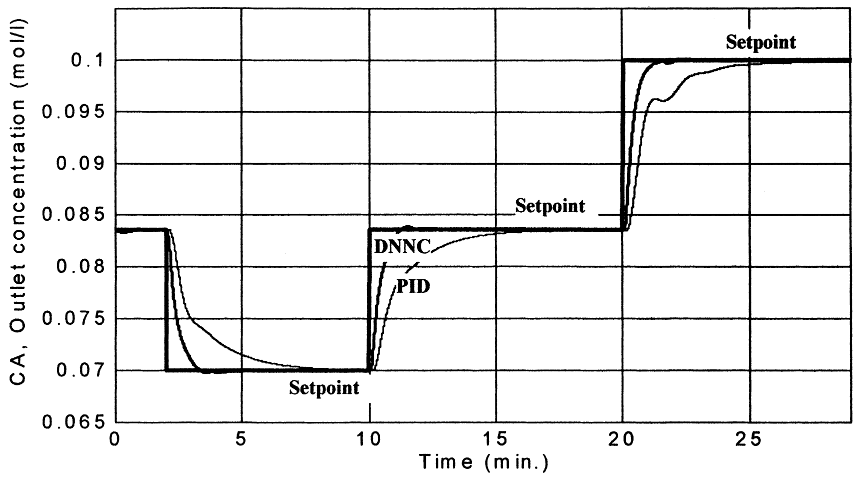

Fig. 8. Setpoint tracking performance of DNNC and PID for CSTR with time varying heat transfer coefficient, [Effect of fouling $\phi_h(t) = (1-\alpha_h t)$ with $\alpha_h = 0.01$].

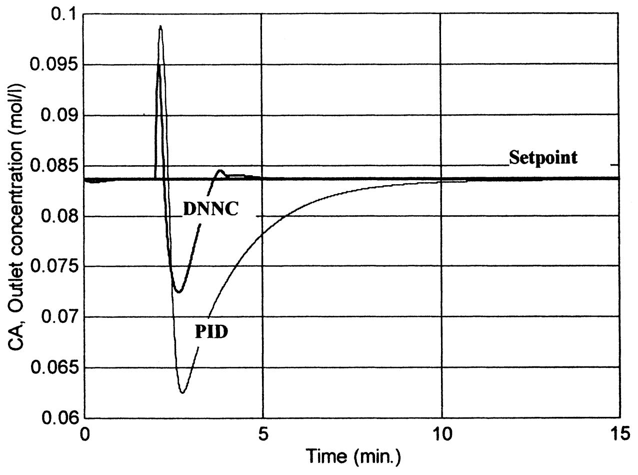

Fig. 9. Disturbance rejection performance of DNNC and PID for +20% change in the feed flow rate of the CSTR with time varying heat transfer coefficient, [Effect of fouling $\phi_h(t) = (1-\alpha_h t)$ with $\alpha_h = 0.01$].

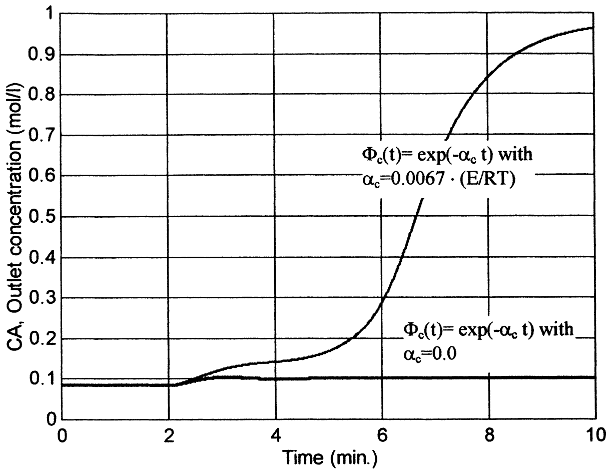

Fig. 11. Comparison between open-loop response of the CSTR with catalyst deactivation and CSTR with constant parameter for +5% step change in the coolant flow rate, q_c.

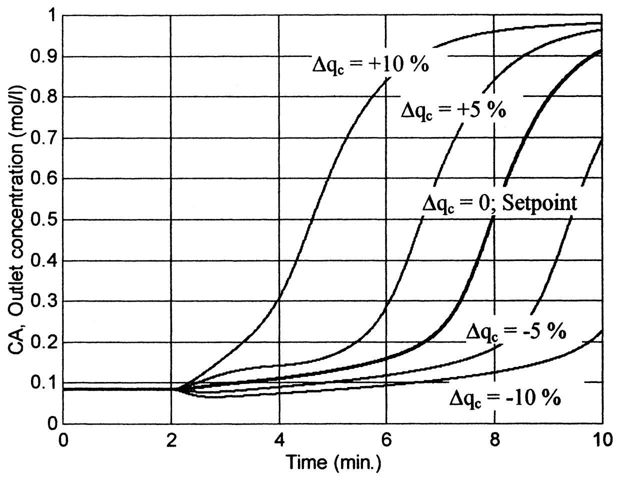

Fig. 10. Open-loop response of the CSTR with catalyst deactivation for step changes in the coolant flow rate, q_c; [Effect of catalyst deactivation $\phi_h(t) = (1-\alpha_h t)$ with $\alpha_h = 0.01$].

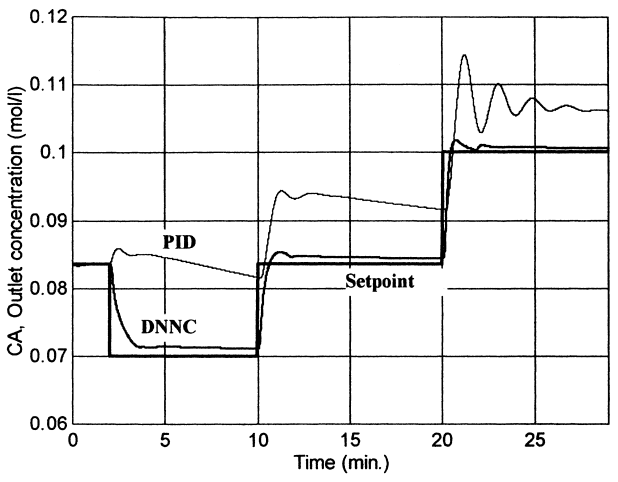

Fig. 12. Setpoint tracking performance of DNNC and PID for CSTR with catalyst deactivation, Effect of catalyst deactivation $\phi_c(t) = \exp(-\alpha_c t)$ with $\alpha_c = 0.0067$. (E/RT).

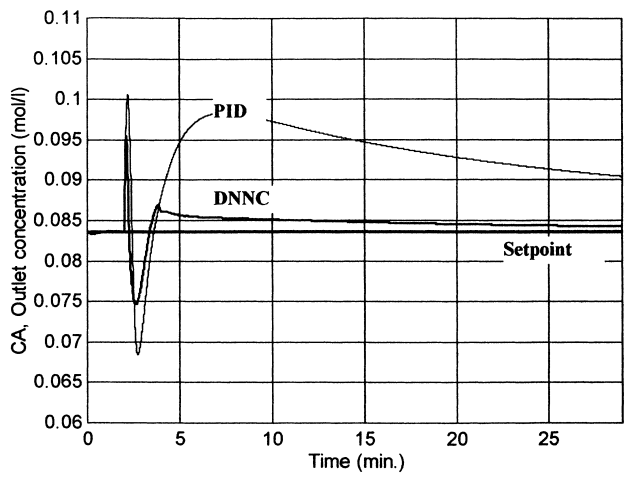

Fig. 13. Disturbance rejection performance of DNNC and PID for +20% change in the feed flow rate of the CSTR with catalyst deactivation, [Effect of catalyst deactivation $\phi_c(t)=\exp(-\alpha_c t)$ with $\alpha_c=0.0067$. (E/RT)].

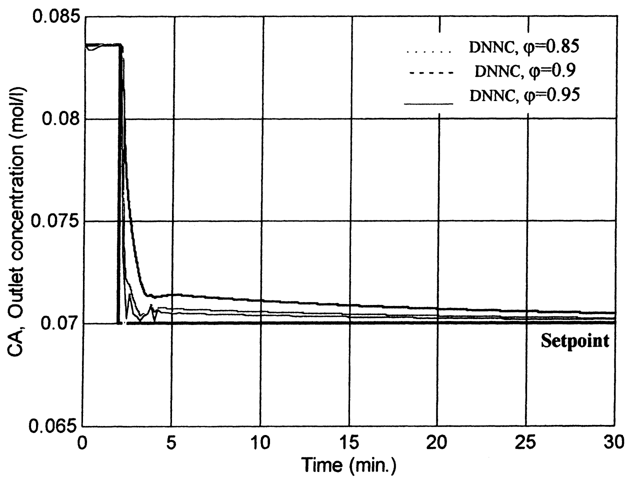

Fig. 14. Setpoint tracking performance of DNNC, Effect of filter constant φ .

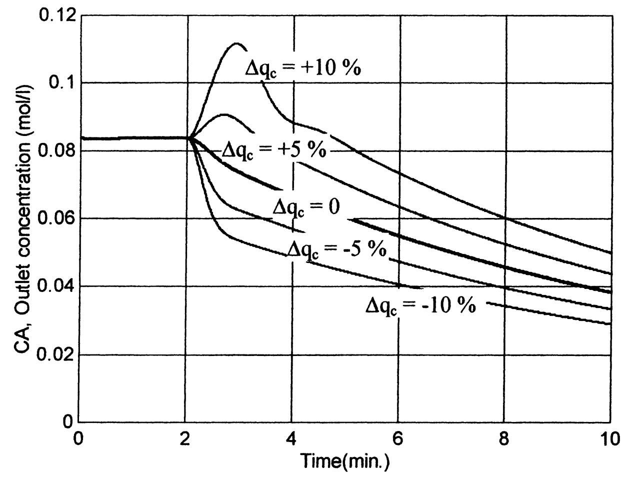

Fig. 15. Open-loop response of the CSTR with catalyst regeneration for step changes in the coolant flow rate, q_c, [Effect of catalyst regeneration $\phi_c(t)=\exp(-\alpha_c t)$ with $\alpha_c=-0.0067$. (E/RT).

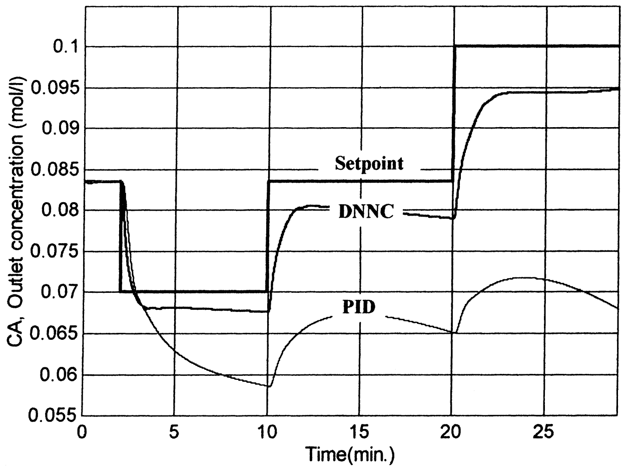

Fig. 16. Setpoint tracking performance of DNNC and PID for CSTR with catalyst regeneration, Effect of catalyst regeneration $\phi_c(t)=\exp(-\alpha_c t)$ with $\alpha_c=0.0067$. (E/RT).

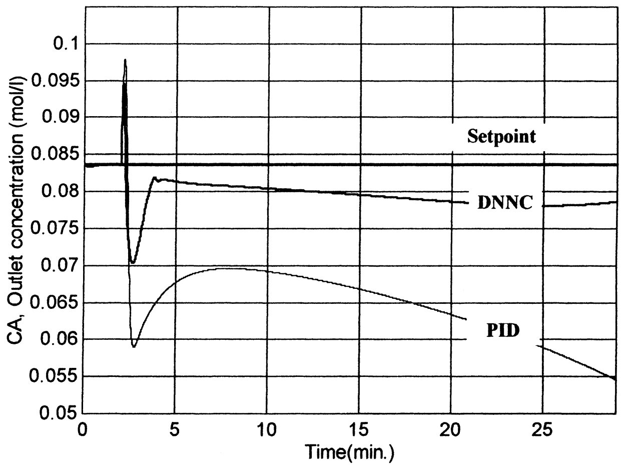

Fig. 17. Disturbance rejection performance of DNNC and PID for 20% change in the feed flow rate of the CSTR with catalyst regeneration, Effect of catalyst regeneration $\phi_c(t)=\exp(-\alpha_c t)$ with $\alpha_c=-0.0067$. (E/RT).

6. Conclusions

This paper provides a detailed case study in which DNNC is applied to the nonisothermal CSTR with time varying parameters including activation energy (i.e., deactivation of catalyst) and heat transfer coefficient (i.e., fouling). DNNC is compared with PID control. Although it is clear that DNNC will perform better than PID, we note that the PID controller performance showed the extreme range of the nonlinearity of the process. This paper represents a preliminary effort to design a simplified neural network-based control approach for a class of nonlinear processes. Therefore, additional work is required for investigation of the effectiveness of this approach for other chemical processes such as batch reactors.

The DNNC strategy differs from previous neural network controllers because the network structure is very simple, having limited nodes in the input and hidden layers. As a result of its simplicity, the DNNC design and implementation are easier than other control strategies such as conventional and hybrid neural networks. In addition to offering a better initialization of network weights, the inverse of the process is exact and does not involve approximation error. DNNC's ability to model nonlinear process behavior does not appear to suffer as a result of its simplicity. This was demonstrated by controlling the nonlinear nonisothermal CSTR with time varying parameters. For the nonlinear, time varying case, the performance of DNNC was compared to the PID control strategy. In comparison with PID control, DNNC showed significant improvement with faster response time toward the setpoint for the servo problem. The DNNC strategy is also able to reject unmodeled disturbances more effectively. DNNC showed excellent performance in controlling the exothermic CSTR in the region where the PID controller failed. It has been shown that the DNNC controller strategy is robust enough to perform well over a wide range of operating conditions.

Appendix A. DNNC Controller design in DMC framework

The DNNC model in IMC framework predicts one future output at each sampling time. For predicting outputs more than one time step in the future, the iteration through the neural network would be required. The predicted output from the neural network can be used at sampling time $k+1$ as input to the neural network at sampling time k. The manipulated inputs Δu must be shifted accordingly. Similar iteration to obtain future prediction is also proposed by Saint-Donat et al. [11]. In this case the controller model is given by the following equations,

$$\Delta \boldsymbol{u} = [[\boldsymbol{W1}^{\mathrm{T}}\,\Gamma 1^{\mathrm{T}}\,\Gamma 1\,\boldsymbol{W1} + \Lambda 1^{\mathrm{T}}\,\Lambda 1]^{-1}\,\boldsymbol{W1}^{\mathrm{T}}\Gamma 1^{\mathrm{T}}\,\Gamma 1\,\mathrm{G}(\boldsymbol{e})], \tag{A.1}$$

where

$$G(\boldsymbol{e}) = F^{-1}\left[\frac{\boldsymbol{e} - \mathrm{B2}\boldsymbol{1}_p}{\mathrm{W2}}\right] - \mathrm{B1}\,\boldsymbol{1}_p - \boldsymbol{y}^{*\mathrm{NN}}(k+1) - \alpha\, y_m^{\mathrm{pred}}(k) \tag{A.2}$$

$$\boldsymbol{e} = y^{\mathrm{set}}(k+1) - \boldsymbol{d}(k) \tag{A.3}$$

with each element of transfer function, F^{-1}, defined by Eq. (46)

$$\boldsymbol{W1} = \begin{bmatrix} w_{1,1} & 0 & 0 & \cdots & 0 \\ w_{1,2} & w_{1,1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{1,M} & w_{1,M-1} & w_{1,N-2} & \cdots & w_{1,1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{1,P} & w_{1,P-1} & w_{1,P-2} & \vdots & w_{1,P-M+1} \end{bmatrix} \tag{A.4}$$

and

$$\boldsymbol{y}^{*NN}(k+1) = \begin{bmatrix} \sum_{j=2}^{N-1} w_{1,j}\,\Delta u(k-j+1) + w_{1,N}u(k-N+1) \\ \sum_{j=3}^{N-1} w_{1,j}\,\Delta u(k-j+2) + w_{1,N}u(k-N+2) \\ \vdots \\ \sum_{j=P+1}^{N-1} w_{1,j}\,\Delta u(k-j+P) + w_{1,N}u(k-N+P) \end{bmatrix} \tag{A.5}$$

$\Delta\boldsymbol{u}(k)$	$M\times 1$ vector of controller output, change in the input (manipulated variable) defined as $u(k)-u(k-1)$,
M	number of manipulated moves into the future,
P	prediction into the future,
N	number of time intervals needed to describe the process dynamics,
$y_m^{\mathrm{pred}}(k)$	current feedback measurement, predicted using neural network,
$y*^{\mathrm{NN}}(k+1)$	output due to input moves up to the present time.
$\boldsymbol{1}_P$	$P\times 1$, vector of ones,
P	$M\le P\le N-1$,
W2	hidden/output weighting factor,
$\boldsymbol{W1}$	$P\times M$ matrix, lower triangular matrix as defined in Eq. (52),
B1	hidden layer bias,
B2	output layer bias,
$\Gamma 1$	$P\times P$ diagonal matrix, tuning parameter,
$\Lambda 1$	$M\times M$ diagonal matrix, tuning parameter.

DNNC in DMC framework provides the same tuning parameters as DMC parameters (N, M, P, γ, λ) for controller design.

Appendix B. Extension of SISO–DNNC to MIMO–DNNC

For MIMO systems with r — output and s — input system, a linear DMC dynamic representation is given by,

$$\boldsymbol{O}\,(k+1) = \boldsymbol{O}^*\,(k+1) + \boldsymbol{A}\,\Delta\boldsymbol{U}\,(k) + \boldsymbol{d}\,(k+1), \tag{B.1}$$

where

$\boldsymbol{O}(k+1)$	r dimensional vector of outputs and each vector with M future prediction for each output,
$\boldsymbol{A}$	$r\times s$ block matrix and each block with matrix of $M\times N$ of unit step response coefficient for the ith time intervals,
$\Delta\boldsymbol{U}(k)$	s dimensioned vector and each vector with N elements of future moves for each manipulated variables (inputs),
$\boldsymbol{d}(k)$	vector of unmodeled factors,
$\boldsymbol{O}_0$	initial condition vector.

with the $\boldsymbol{A}$ matrix define as,

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} & \dots & \boldsymbol{A}_{1,s}\\ \boldsymbol{A}_{2,1} & \boldsymbol{A}_{2,2} & \dots & \boldsymbol{A}_{2,s}\\ \vdots & \vdots & \vdots & \vdots\\ \boldsymbol{A}_{r,1} & \boldsymbol{A}_{r,2} & \dots & \boldsymbol{A}_{r,s}\end{bmatrix} \tag{B.2}$$

such that A_{ij} contains all the *ij* coefficient in matrices a_l, $l=1$, M (refer to $\boldsymbol{A}$ matrix for the DMC model for SISO case as described in Chapter 2 Ref. [26]).

The following optimization method is used to find the N future moves $\Delta I(k),\dots,\Delta I(k+N-1)$,

$$\underset{\Delta I}{\text{Min}}\left[\sum_{i=1}^{p}\lambda^2(i)[Y^{\text{sp}}(k+i)-Y(k+i)]^2 + \sum_{j=1}^{M}\gamma^2(j)[\Delta U(k+M-j)]^2\right]. \tag{B.3}$$

The solution of such least-squares problems is given by

$$\Delta\boldsymbol{I}\,(k)=\left[\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{\Gamma}\,\boldsymbol{A}+\boldsymbol{\Lambda}^{\mathrm{T}}\boldsymbol{\Lambda}\right]^{-1}\boldsymbol{A}^{\mathrm{T}}\boldsymbol{\Gamma}^{\mathrm{T}}\boldsymbol{\Gamma}\quad[\boldsymbol{e}\,(k+1)], \tag{B.4}$$

where

$$\Lambda=\text{diag}(\lambda_1\lambda_1\dots\lambda_1\lambda_2\underset{|\leftarrow M\rightarrow|}{\lambda_2\dots\lambda_2}\dots\lambda_s\lambda_s\dots\lambda_s), \tag{B.5}$$

λ_{s} = move suppression parameter,

$$\Gamma=\text{diag}(\gamma_1\gamma_1\dots\gamma_1\gamma_2\underset{|\leftarrow P\rightarrow|}{\gamma_2\dots\gamma_2}\dots\gamma_r\gamma_r\dots\gamma_r). \tag{B.6}$$

γ_r = output weighing parameter.

By analogy, the DNNC controller equation for the MIMO case is given by the following equations:

$$\Delta\boldsymbol{u}=\left[\boldsymbol{W1}^{\mathrm{T}}\,\Gamma 1^{\mathrm{T}}\Gamma 1\,\boldsymbol{W1}+\Lambda 1^{\mathrm{T}}\Lambda 1\right]^{-1}\boldsymbol{W1}^{\mathrm{T}}\,\Gamma 1^{\mathrm{T}}\Gamma 1\quad G(\boldsymbol{e}), \tag{B.7}$$

where

$$\mathbf{W1}=\begin{bmatrix}\boldsymbol{W}_{1,1} & \boldsymbol{W}_{1,2} & \dots & \boldsymbol{W}_{1,s}\\ \boldsymbol{W}_{2,1} & \boldsymbol{W}_{2,2} & \dots & \boldsymbol{W}_{2,s}\\ \vdots & \vdots & \vdots & \vdots\\ \boldsymbol{W}_{r,1} & \boldsymbol{W}_{r,2} & \dots & \boldsymbol{W}_{r,s}\end{bmatrix} \tag{B.8}$$

with

$$\Lambda 1=\text{diag}(\lambda_1\lambda_1\dots\lambda_1\lambda_2\underset{|\leftarrow M\rightarrow|}{\lambda_2\dots\lambda_2}\dots\lambda_s\lambda_s\dots\lambda_s) \tag{B.9}$$

λ_{s} = move suppression parameter,

$$\Gamma 1=\text{diag}\,(\gamma_1\gamma_1\dots\gamma_1\gamma_2\underset{|\leftarrow P\rightarrow|}{\gamma_2\dots\gamma_2}\dots\gamma_r\gamma_r\dots\gamma_r) \tag{B.10}$$

γ_r = output weighing parameter, such that W_{ij} contains all the *ij* coefficients in matrices w_l, $l=1,M$ (refer to W1 matrix for the DNNC model for SISO case as described in Chapter 2 of Ref. [26]). The rest of the equations for the MIMO–DNNC can be derived by analogy and comparing to MIMO–DMC equations. Details of MIMO–DMC is available throughout the literature.

References

[1] G. Cybenko, Approximation by superposition of a sigmoidal function, Math. Control Sig. System 2 (1989) 303.

[2] R. Hecht-Nielsen, Theory of backpropagation neural networks, in: IEEE Proceedings of the International Conference on Neural Networks, Washington DC, 1989, p. I-593.

[3] C.D. Pschogios, L.H. Ungar, A hybrid neural network-first principles approach to process modeling, AIChE J. 10 (1992) 1499.

[4] K.S. Narendra, K. Parthasarathy, Identification and control of dynamical systems, using neural networks, IEEE Trans. Neural Networks 1 (1990) 4.

[5] J.F. Pollard et al., Process identification using neural networks, Comp. Chem. Eng. 4 (1992) 253.

[6] N.V. Bhat et al., Modeling chemical process systems via neural computation, IEEE Control System Mag. (1990) 24.

[7] S. Chen, S.A. Billings, P.M. Grant, Nonlinear systems identification using neural networks, Int. J. Control 51(6) (1990) 1191.

[8] M. Nikravesh, A.E. Farell, Modeling and controlling nonlinear chemical processes using hybrid neural network dynamic matrix control algorithm (NNDMC), AIChE Annual Meeting, St. Louis, 1993, paper 213b8.

[9] C.D. Pschogios, L.H. Ungar, Direct and indirect model based control using artificial neural networks, Ind. Eng. Chem. Res. 30 (1991) 2564.

[10] M. Lee, S. Park, A new scheme combining neural feedforward control with model predictive control, AIChE J. 38(2) (1992) 193–200.

[11] J. Saint-Donat, N. Bhat, T.J. McAvoy, Neural net based model predictive control, Int. J. Control 54(6) (1991) 1453.

[12] E. Nahas, M. Henson, D. Seborg, Nonlinear internal model control strategy for neural network models, Comp.Chem. Eng. 16(12) (1992) 1039.

[13] M. Nikravesh et al., Identification and control of industrial-scale processes via neural networks, presented at Chemical Process Control V, Tahoe City, CA, 5–12 January 1996.

[14] M. Soroush, C. Kravaris, Discrete-time nonlinear controller synthesis by input/output linearization, AIChE J. 38 (1992a) 1923–1945.

[15] M. Soroush, C. Kravaris, Feedforward/feedback control of discrete-time nonlinear systems, AIChE Annual Meeting, 1–6 November 1992, Miami, Paper 126e, 1992b.

[16] M.M. Gupta, Fuzzy logic and fuzzy systems: recent developments and directions, Biennial Conference of the North American Fuzzy Information Processing-NAFIPS, Berkeley, CA, 1996, pp. 155–159.

[17] M.M. Gupta, D.H. Rao, Dynamic neural units in the control of linear and nonlinear systems, International Joint Conferences on Neural Networks, vol. II, Baltimore, 7–11 June 1992, pp. 100–105.

[18] M. Nikravesh, A.E. Farell, T.G. Stanford, Model identification of nonlinear time variant processes via artificial neural network, Comp. Chem. Eng. 20(11) (1995) 1277–1290.

[19] N. Bhat, T.J. McAvoy, Use of neural nets for dynamic modeling and control of chemical process systems, Comp. Chem. Eng. 14(4/5) (1990) 573.

[20] E. Hernandez, Y. Arkun, ACC Conference 1990, p. 2454.

[21] L.G. Kraft, D.P. Campagna, A comparison between CMAC neural network control and two traditional adaptive control systems, IEEE Control Systems Mag., 1992, p. 36.

[22] J.D. Cooper et al., Comparing two neural networks for pattern based adaptive process control, AIChE J. 38 (1) (1992) 42.

[23] C.E. Garcia, D.M. Pret, M. Morari, Model predictive control — a survey, Automatica 25 (1989) 335.

[24] A. Uppal, W.H. Ray, On the dynamic behavior of continuos stirred tank reactors, Chem. Eng. Sci. 29 (1974) 967.
[25] L.C. Limqueco, J.C. Kantor, Nolinear output feedback control of an exothermic reactor, Comp. Chem. Eng. 14 (1990) 427.
[26] M. Nikravesh, Dynamic neural network control, Ph.D. Dissertation, University of South Carolina, Columbia, SC, 1994.
[27] C.R. Cutler, Dynamic matrix control — a computer control algorithm, AIChE 86th National Meeting, Houston, TX, 1979.
[28] C.R. Cutler, B.L. Ramaker, Proceedings of the Joint Automation and Control Conference, San Francisco, CA, WP5-B 1980 and AIChE 86th National Meeting, Houston, TX, 1979.
[29] M. Morari, E. Zafiiriou, Robust Process Control, Prentice-Hall, Englewood Cliffs, NJ, 1989.
[30] A. Isidori, Nonlinear Control Systems: An Introduction, 2nd ed., Springer, New York, 1989.
[31] A. Henson, D.E. Seborg, AIChE J. 37(7) (1991) 1065.
[32] C.E. Garcia, M. Morari, Internal model control. 1. A unifying review and some new results, Ind. Eng. Chem. Process Des. 21(1982) 308.
[33] P.B. Deshpande, Multi-variable Control Methods, ISA, 1988.
[34] P.B. Deshpande, Computer Process Control with Advanced Control Application, ISA, 1988.
[35] L. Jin, M.M. Gupta, P.N. Nikiforuk, 13th Triennial World Congress, vol. F, San Francisco, USA, 1996, pp. 187–192.
[36] K. Ogata, Discrete Time Control Systems, Prentice-Hall, Englewood Cliffs, NJ, 1987.
[37] R. Langari, W. Li, Analysis and efficient implementation of fuzzy logic control algorithms, Biennial Conference of the North American Fuzzy Information Processing NAFIPS, Berkeley, CA, 1996, pp. 1–4.
[38] J.P. LaSalle, The Stability and Control of Discrete Processes, Springer, New York, 1986.
[39] L. Jin, P.N. Nikiforuk, M.M. Gupta, Absolute stability conditions for discrete-time recurrent neural networks, IEEE Trans. Neural Networks 5 (1994) 945–955.
[40] L. Jin, P.N. Nikiforuk, M.M. Gupta, Dynamics and stability of multi-layered recurrent neural networks, Proceedings of the 1993 IEEE International Conference on Neural Networks (ICNN‘93), vol. II, 1993, pp. 1135–1140.
[41] M. Nikravesh, A.E. Farell, T.G. Stanford, Dynamic neural network control for nonlinear systems: optimal neural network structure and stability analysis, Chem. Eng. J. 68 (1997) 41–50.
[42] E. Hernandez, Y. Arkun, Study of the control relevant properties of backpropagation neural net models of nonlinear dynamical systems, Comp. Chem. Eng. 16(4) (1992) 227–240.
[43] R.H. Perry, C.H. Chilton, Chemical Engineering Handbook, 5th ed., McGraw-Hill, New York, 1973.
[44] G.F. Froment, K.B. Bischoff, Chemical Reactor Analysis and Design, 2nd ed., Wiley, New York, 1990.