A modified Hammerstein modeling by the differential evolution algorithm

Wei-Der Chang

wdchang@stu.edu.tw

Shu-Te University

Research Article

Keywords: Bilinear neural network, Recursive digital system, Hammerstein model, Differential evolution algorithm, System modeling

Posted Date: February 7th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2548758/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Signal, Image and Video Processing on May 29th, 2024. See the published version at https://doi.org/10.1007/s11760-024-03218-w.

A modified Hammerstein modeling by the differential evolution algorithm

Wei-Der Chang

Department of Computer and Communication, Shu-Te University
Kaohsiung 824, Taiwan
E-mail: wdchang@stu.edu.tw

Abstract

This paper focuses on the nonlinear system modeling based on using a modified Hammerstein system model. The proposed Hammerstein structure is composed of a bilinear neural network (BNN) and a recursive digital system in the cascaded form. The former is taken to be the nonlinear function part of the Hammerstein model, and the latter is used as the linear dynamic subsystem. The BNN is then constructed by the bilinear digital system and the recurrent neural network, which already possesses a satisfactory modeling capacity. To update all the adjustable parameters within the proposed Hammerstein model, a popular and powerful evolutionary computation called the differential evolution (DE) is utilized so that the model output can be very close to the actual nonlinear system output. Finally, a simulated nonlinear chemical process system, continuously stirred tank reactor (CSTR), is illustrated with the modeling phase and testing phase. Some numerical results as compared with a different method from subject literature are provided to show the feasibility of the proposed method and its good modeling.

***Keywords*:** Bilinear neural network; Recursive digital system; Hammerstein model; Differential evolution algorithm; System modeling.

1. Introduction

For most of practical systems, their properties both static and dynamic are nonlinear and complicated in nature. It is rather hard to set up the mathematical model correspondingly to the actual system by means of the linear model such as a differential equation and/or a difference equation. In fact, a collection of system input-output pairs is the only information that is available in most real situations. Thus, question of how to establish a suitable model structure for modeling such systems remains to be solved. Another problem that occurs is then how to design these adjustable parameters inside the proposed model when the model has been developed. To tackle the modeling problem, a lot of nonlinear structure models have recently been reported such as the neural network model, fuzzy system model, Volterra series model, bilinear series model, Hammerstein model etc. For adjusting model parameters, the general gradient-based method is often adopted. But, one main drawback is that the system solution derived by such gradient descent method is likely to fall into the local optima around the initial condition.

The focus of this paper is the modified Hammerstein model combined with the differential evolution (DE) algorithm. This kind of Hammerstein model has widely been used to solve the nonlinear system modeling problem [1]-[7]. It is composed of two main blocks: a nonlinear static block and a linear dynamic block. These two system blocks are cascaded. In the traditional fashion, the first block refers to the static nonlinear function that can be constructed by, for instance, polynomial functions, sinusoidal functions, and dead-zone nonlinearities. The latter is a recursive digital system alternatively referred to as an infinite impulse response (IIR) digital filter. In [2], a Hammerstein model consisting of a functional link artificial neural network (FLANN) in cascade with an adaptive infinite impulse response filter has been

developed. A cuckoo search algorithm was employed to train the proposed model parameters. In [3], a novel method to identify parallel Wiener-Hammerstein systems using input-output data only was presented, and this method was validated on real-world measurements by a custom built parallel Wiener-Hammerstein test systems. In [5], a hierarchical gradient parameter estimation algorithm was proposed for Hammerstein nonlinear systems by means of the key term separation principle. Simulation results show the effectiveness of the proposed algorithm. This paper develops a new modified Hammerstein model in which the bilinear neural network (BNN) followed by the recursive digital system is considered. Moreover, the BNN is then constructed by the bilinear digital system and the recurrent neural network.

To design the adjustable model parameters, one of the optimal algorithms called DE is employed, instead of the commonly used gradient descent scheme. The DE was initially proposed by Storn and Price in 1997 and has been proven to be a powerful but simple means for solving optimization problem [8]. This kind of algorithm has a good chance of converging to the global solution of the optimized problem because it is a population-based algorithm with multiple direction searches. Basically, there are three important operations in the DE algorithm to achieve optimized solution, including mutation, crossover, and selection, which seems to be similar to those in present genetic algorithms (GAs). Application of DE has been demonstrated on a variety of engineering optimization problems, such as: the transmission expansion planning (TEP) problem [9], mobile ad hoc networks (MANETs) [10], real power loss minimization [11], underwater glider path planning [12], wireless sensor networks [13], and others [14]-[17]. In addition, some new variants of the DE algorithm to improve the search efficiency or to overcome special problems were also presented in [18]-[22].

This paper proposes a modified version of Hammerstein model where the BNN is used as the nonlinear static block of the model and the recursive digital system as the linear dynamic block. Such a BNN modeling structure consisting of the bilinear digital system and the recurrent neural network was initially developed in [23]. In the BNN model, the terms corresponding to cross-products of input and output signals within the bilinear system are taken as inputs to the recurrent neural network. This kind of model can be regarded as an independent mathematical model, as it provides a good modeling performance on its own. Nevertheless, in order to obtain a more flexible model for identification, a new modified Hammerstein model is presented that combines such a BNN model with a recursive digital system. Moreover, the DE is utilized to find values of all adjustable parameters contained in the proposed Hammerstein model, so that the model output is capable of closely approximating the actual nonlinear system output. The remainder of this paper is organized as follows. A modified version of Hammerstein model consisting of the BNN and the recursive digital system is introduced in Section 2. In Section 3, a clear and complete description of the DE algorithm is addressed and the design steps for DE-based Hammerstein modeling are also proposed. Section 4 will illustrate the proposed method by a highly nonlinear chemical process of continuously stirred tank reactor (CSTR). Simulation results in comparison with other method taken from subject literature demonstrate the superiority of the proposed scheme. Finally, some conclusions and future research directions are given in Section 5.

2. A modified Hammerstein model

- **A general Hammerstein model**

The general Hammerstein model is composed of two system blocks including the

nonlinear static block and the linear dynamic block in the cascaded form. Fig. 1 shows the block diagram where $u[n]$ is the external input signal, $x[n]$ is the output of nonlinear static block, also referred to as an intermediate signal of the model, $y_m[n]$ is the model final output, $f(\cdot)$ represents the nonlinear static function, and $H(z^{-1})$ represents the transfer function of linear dynamic system in which z^{-1} means a unit delay operator. We can further describe the Hammerstein model by

$$x[n] = f(u[n]; S), \tag{1}$$

$$y_m[n] = H(z^{-1})x[n], \tag{2}$$

where *S* is a collection of all adjustable parameters contained in the nonlinear function. In the proposed model, the BNN is used as the nonlinear static block and the recursive digital filter as the linear dynamic block, respectively. Fig. 2 displays the proposed Hammerstein model structure. Adjustable parameters within the proposed model totally include all weights and thresholds of BNN and coefficients of recursive digital system. These model parameters are necessary to be adjusted according to the measured input-output data obtained from the actual nonlinear system which will be modeled. Both BNN and recursive filter are further explained in detail in subsequent sections.

- **Bilinear Neural network (BNN)**

BNN is constructed by the bilinear digital system and the recurrent neural network [23]. All cross-product terms of input and output signals contained in such a bilinear system are taken to be the input vector to the recurrent neural network. Generally, the input-output bilinear digital system can be expressed by

$$x[n] = \sum_{k=1}^{N_x} a_k x[n-k] + \sum_{k=0}^{N_u} b_k u[n-k] + \sum_{k_1=0}^{N_u}\sum_{k_2=1}^{N_x} c_{k_1,k_2} u[n-k_1]x[n-k_2]$$

$$= a_1 x[n-1] + a_2 x[n-2] + \cdots + a_{N_x} x[n-N_x]$$

$$+ b_0 u[n] + b_1 u[n-1] + \cdots + b_{N_u} u[n-N_u]$$

$$+ c_{0,1} u[n]x[n-1] + c_{0,2} u[n]x[n-2] + \cdots + c_{0,N_x} u[n]x[n-N_x] + \cdots$$

$$+ c_{N_u,1} u[n-N_u]x[n-1] + c_{N_u,2} u[n-N_u]x[n-2] + \cdots$$

$$+ c_{N_u,N_x} u[n-N_u]x[n-N_x], \tag{3}$$

where $u[n]$ and $x[n]$ are the input and output signals, a_k and b_k represent the coefficients of the linear part, c_{k_1,k_2} are the coefficient of cross-product terms of input and output signals, N_x is the system order representing the number of past outputs, and N_u is the number of past input signals. Eq. (3) can further be rewritten in the vector form for brevity as

$$x[n] = AX^T + BU^T + CW^T, \tag{4}$$

where *A*, *B*, and *C* represent coefficient vectors defined by

$$A = [a_1, a_2, \cdots, a_{N_x}],$$

$$B = [b_0, b_1, \cdots, b_{N_u}], \tag{5}$$

$$C = [c_{0,1}, c_{0,2}, \cdots, c_{N_u,N_x}],$$

and vectors *X*, *U*, and *W* denote the past output signals, present and past input signals, and all cross-product terms of input and output signals, respectively, and are given by

$$X = [x[n-1], x[n-2], \cdots, x[n-N_x]],$$

$$U = [u[n], u[n-1], \cdots, u[n-N_u]], \tag{6}$$

$$W = [u[n]x[n-1], u[n]x[n-2], \cdots, u[n-N_u]x[n-N_x]].$$

Furthermore, Eq. (4) can be more compactly expressed by

$$x[n] = DZ^T, \tag{7}$$

where $D = [A \quad B \quad C]$ is a collection of all system coefficients and $Z = [X \quad U \quad W]$ is the vector collecting all system input-output signals. In the BNN structure, the vector *Z* plays an important role and it is taken as the input to the neural network. The vector length of both *D* and *Z* is easily calculated by Eq. (8)

$$N = N_x + (N_u + 1) + (N_u + 1)N_x = (N_u + 1)(N_x + 1) + N_x. \tag{8}$$

Moreover, the vector *Z* is redefined again for a clear representation as Eq. (9)

$$\begin{aligned} Z &= [z_1, z_2, \cdots, z_N] \\ &= [x[n-1], x[n-2], \cdots, x[n-N_x], u[n], u[n-1], \cdots, u[n-N_u], \\ &\qquad u[n]x[n-1], u[n]x[n-2], \cdots, u[n-N_u]x[n-N_x]] \end{aligned} \tag{9}$$

Fig. 3 shows the overall structure of the BNN where the vector *Z* of Eq. (9) is used to be the input signal of the network, *N* and *M* are the number of neurons in the input and hidden layers, respectively, w_zh_{ij} denotes the weight between the *i*th input neuron with the *j*th hidden neuron, w_hx_j is the weight from the *j*th hidden neuron to the output neuron, all for $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, M$, and $x[n]$ is the final output of the BNN. Notice that the delay signals of $x[n]$ are requested to feed back to the input layer forming a recurrent neural network as addressed in Eq. (9).

In the hidden layer, each neuron has the following mathematical operations:

$$net\_h_j = \sum_{i=1}^{N} w\_zh_{ij} \cdot z_i - \theta\_h_j, \tag{10}$$

$$h_j = \frac{e^{net\_h_j} - e^{-net\_h_j}}{e^{net\_h_j} + e^{-net\_h_j}}, \tag{11}$$

where net_h_j, θ_h_j, and h_j are the internal state, threshold, and output signal of the *j*th hidden neuron, respectively, for $j = 1, 2, \cdots, M$, and the activation function is the hyperbolic tangent function given by Eq. (11). All output signals in the hidden

layer are further delivered to the output neuron via the weights using the following equations

$$net\_x = \sum_{j=1}^{M} w\_hx_j \cdot h_j - \theta\_x, \tag{12}$$

$$x[n] = net\_x, \tag{13}$$

where net_x, θ_x, and $x[n]$ are the internal state, threshold, and final output of the BNN, respectively, and the linear activation function here is utilized as given by Eq. (13). Based on the BNN structure of Fig. 3, the total of all adjustable parameters including weights and thresholds contained in the network can be evaluated by

$$P = N \times M + M + M + 1 = M(N+2) + 1. \tag{14}$$

Again, for convenience let the vector *S* be a collection of all adjustable parameters in the BNN as

$$\begin{aligned} S &= [s_1, s_2, \cdots, s_P] \\ &= [w\_zh_{11}, w\_xh_{12}, \cdots, w\_xh_{1M}, w\_zh_{21}, \cdots, w\_zh_{NM}, \\ &\qquad \theta\_h_1, \theta\_h_2, \cdots, \theta\_h_M, w\_hx_1, w\_hx_2 \cdots, w\_hx_M, \theta\_x]. \end{aligned} \tag{15}$$

- **Recursive digital system**

In this subsection, the recursive digital system is introduced in detail. In the developed Hammerstein model, the recursive digital system is cascaded with the BNN, therefore the output $x[n]$ from the BNN is considered as the input signal to the recursive digital system as shown in Fig. 2. Consider the following recursive digital system described by the transfer function

$$H(z^{-1}) = \frac{Y_m(z^{-1})}{X(z^{-1})} = \frac{g_0 + g_1 z^{-1} + g_2 z^{-2} + \cdots + g_{N_g} z^{-N_g}}{1 + d_1 z^{-1} + d_2 z^{-2} + \cdots + d_{N_d} z^{-N_d}} = \frac{\sum_{k=0}^{N_g} g_k z^{-k}}{1 + \sum_{k=1}^{N_d} d_k z^{-k}}, \tag{16}$$

where N_d represents the system order, N_g is the number of past inputs, d_k and g_k are system coefficients. Furthermore, converting Eq. (16) into time domain difference equation yields

$$y_m[n] = -d_1 y_m[n-1] - d_2 y_m[n-2] - \cdots - d_{N_d} y_m[n-N_d] + g_0 x[n] + g_1 x[n-1] + \cdots + g_{N_g} x[n-N_g]$$

$$= -\sum_{k=1}^{N_d} d_k y_m[n-k] + \sum_{k=0}^{N_g} g_k x[n-k], \quad (17)$$

where $x[n]$ is the input signal generated by the BNN and $y_m[n]$ is the output signal of the recursive digital system, i.e., the final output of the developed Hammerstein model. The recursive difference equation diagram for Eq. (17) can simply be depicted in Fig. 4. In the recursive digital system, d_k and g_k are the system coefficients that will be designed together with the parameter vector *S* of Eq. (15). Thus, let $D = [d_1, d_2, \cdots, d_{N_d}]$ and $G = [g_0, g_1, \cdots, g_{N_g}]$ be two coefficient vectors with N_d and $N_g + 1$ parameters, respectively. This means that, in total, there are $N_d + N_g + 1$ adjustable parameters in the recursive digital system.

- **A modified Hammerstein model**

The combination of BNN and recursive digital system forms a modified Hammerstein system with input signal $u[n]$ and output signal $y_m[n]$ as shown in Fig. 2. In the proposed model, there are in total $R = P + N_d + N_g + 1$ designed parameters in which *P* is the number of adjustable parameters of the BNN and $N_d + N_g + 1$ is the corresponding number of coefficients of the recursive digital system. Moreover, for the usage of the DE algorithm let the parameter vector $\Theta = [\theta_1, \theta_2, \cdots, \theta_R]$ be a

collection of all deigned parameters contained in the modified Hammerstein model as follows

$$\begin{aligned}\Theta &= [\theta_1, \theta_2, \cdots, \theta_R] \\ &= [S, D, G] \\ &= [w\_zh_{11}, w\_zh_{12}, \cdots, \theta\_x, d_1, d_2, \cdots, d_{N_d}, g_0, g_1, \cdots, g_{N_g}],\end{aligned} \tag{18}$$

where the subscript R denotes the number of all design parameters. This defined vector Θ will be updated by the DE algorithm.

3. Differential evolution and its applications to Hammerstein modeling

The algorithm chosen for updating all of adjustable parameters is the DE algorithm and it has been shown to be effective for solving such optimization problems. Like most evolutionary methods, this algorithm is also population-based. Hence, the DE begins with generating an initial population containing a large number of parameter vectors as described in Eq. (18). To achieve optimization purpose, it consists of three principal operations: mutation, crossover, and selection to evolve these parameter vectors inside the population. They are explained in detail below [8][23].

- **Mutation**

In the mutation operation, three different parameter vectors named as Θ_α, Θ_β, and Θ_γ are randomly chosen from the population and they are mutually mutated as follows

$$V = \Theta_\alpha + f_s \cdot \left(\Theta_\beta - \Theta_\gamma\right), \tag{19}$$

where the derived vector $V = [v_{1,} v_2, \cdots v_R]$ is called the mutant vector and f_s is the mutation constant factor to adjust the amplification of the differential variation

$\Theta_\beta - \Theta_\gamma$.

- **Crossover**

Consider another target vector $\Theta = [\theta_1, \theta_2, \cdots, \theta_R]$ that is crossed with the mutation vector *V*. It needs a set of random numbers $\{r_1, r_2, \cdots, r_R\}$ where $r_i \in [0, 1]$ for $i = 1, 2, \cdots, R$ to derive another set of binary numbers $\{m_1, m_2, \cdots, m_R\}$ by using Eq. (20)

$$m_i = \begin{cases} 1, & if\ r_i < CR, \\ 0, & otherwise, \end{cases} \quad \text{for } i = 1, 2, \cdots, R \tag{20}$$

where CR is called the crossover rate chosen from the interval $[0, 1]$ and it is always set to 0.5 for the common situation. By means of this binary set $\{m_1, m_2, \cdots, m_R\}$, the so-called trial vector $T = [t_1, t_2 \cdots, t_R]$ can be obtained according to Eq. (21)

$$t_i = \begin{cases} \theta_i, & if\ m_i = 1 \\ v_i, & if\ m_i = 0 \end{cases}, \text{ for } i = 1, 2, \cdots, R. \tag{21}$$

It can be seen that the trial vector *T* is an outcome of crossing Θ with *V*.

- **Selection**

Briefly, the selection operation is to retain the better parameter of the two, i.e. the target vector Θ and the trail vector *T*. To evaluate the performance of each parameter vector, the following cost function is adopted for Hammerstein modeling

$$CF = \sum_{n=0}^{NS} e^2[n] = \sum_{n=0}^{NS} [y[n] - y_m[n]]^2, \tag{22}$$

where *NS* is the number of samples, *e* is the error signal between the actual output *y* and the developed Hammerstein model output y_m. Based on using Eq. (22), the

selection is performed. If $CF(T) < CF(\Theta)$, it means that the trial vector T is superior in terms of modeling fit to the target vector Θ, then the vector T survives and takes the place of Θ. Otherwise, this trial vector T is discarded and the target vector Θ remains in the population. Notice that one generation is defined when the above three operations are executed. The algorithm stops when the assigned allowable number of generations is attained.

- **DE-based design steps for Hammerstein modeling**

In this paper, the proposed Hammerstein modeling architecture for the actual nonlinear system is displayed in Fig. 5 where the DE algorithm is employed to adjust the model parameters, so that the error signal between the actual output and the model output can be minimized. The following lists the complete design steps.

Data: A set of input-output pairs of a real nonlinear system $Q = \{(u[n],\ y[n]) \mid n = 0, 1, 2, \cdots, SN\}$, the BNN structure with N-input neurons, M-hidden neurons, and a single output neuron, the order N_d and N_g of digital recursive system of Eq. (17), population size PS, mutation constant factor f_s in Eq. (19), and allowable number of generations G_num.

Goal: Construct a modified Hammerstein model consisting of the BNN and recursive digital system to identify the actual nonlinear system by the DE algorithm.

Create an initial population with PS parameter vectors randomly chosen from the interval $[-1, 1]$.

for $j = 1$ to G_num

for $i = 1$ to PS

Calculate the corresponding cost function $CF(\Theta_i)$ of the target vector Θ_i using Eq. (22).

Obtain the mutation vector *V* from Eq. (19).

Derive a set of binary sequences $\{m_1, m_2, \cdots, m_R\}$ by using Eq. (20).

Perform the crossover operation on the target vector Θ_i and the mutation vector *V* to obtain the trial vector *T* using Eq. (21).

Calculate the cost function $CF(T)$ corresponding to *T* using Eq. (22).

Perform the selection operation on Θ_i and *T*. If $CF(T) < CF(\Theta_i)$, then $\Theta_i^{new} = T$, otherwise $\Theta_i^{new} = \Theta_i$.

End

for $i = 1$ to *PS*

$\Theta_i = \Theta_i^{new}$.

End

End

4. Modeling simulation: a chemical case of CSTR

A case study for modeling a chemical process system demonstrates applicability of the proposed method. The chemical process used is the continuously stirred tank reactor (CSTR) that is a highly nonlinear and complex system expressed by the following dynamical equations [24]

$$\dot{x}_1 = -x_1 + D_a(1-x_1)\exp(\frac{x_2}{1+x_2/\varphi}),$$

$$\dot{x}_2 = -(1+\delta)x_2 + BD_a(1-x_2)\exp(\frac{x_2}{1+x_2/\varphi}) + \delta u, \quad (23)$$

$y = x_2$,

where x_1 and x_2 are the system states and represent the dimensionless reactant concentration and the reactor temperature, respectively, *u* and *y* denote the system input and output (reactor temperature), other system parameters including D_a, φ, *B*, and δ are explained in Tab. 1. Converting Eq. (23) into a discrete-time form by a simple Euler's discretizton yields

$$x_1[n+1] = x_1[n] + T_s(-x_1[n] + D_a(1 - x_1[n])\exp(\frac{x_2[n]}{1 + x_2[n]/\varphi})),$$

$$x_2[n+1] = x_2[n] + T_s(-(1+\delta)x_2[n] + BD_a(1 - x_2[n])\exp(\frac{x_2[n]}{1 + x_2[n]/\varphi}) + \delta u[n]), \quad (24)$$

$$y[n] = x_2[n],$$

where T_s is the sampling interval of 0.2s for simulations.

For a fair comparison the parameter setting used in the BNN and DE algorithm are the same with Ref. [23] listed in Tab. 2. Simulation results consisting of two phases: modeling and testing phases are provided to verify the identification performance. Firstly, in the modeling phase the input signal $u[n]$ is generated from the interval $[-1, 1]$ randomly and uniformly for $n = 0, 1, \cdots, 100$. Two different input signals are utilized in the testing phase; one is the uniformly generated random number chosen from $[-1, 1]$, the other is Gaussian random number with mean $\mu = 0.0$ and variance $\sigma^2 = 0.5$, all for $n = 0, 1, \cdots, 200$. In addition, three different architectures for the digital recursive system are examined in the modified Hammerstein model, as follows.

- Case 1: $N_d = 3$ and $N_g = 1$,

  $y_m[n] = -d_1 y_m[n-1] - d_2 y_m[n-2] - d_3 y_m[n-3] + g_0 x[n] + g_1 x[n-1]$,

- Case 2: $N_d = 4$ and $N_g = 1$,

$$y_m[n] = -d_1 y_m[n-1] - d_2 y_m[n-2] - d_3 y_m[n-3] - d_4 y_m[n-4] + g_0 x[n] + g_1 x[n-1],$$

- Case 3: $N_d = 4$ and $N_g = 2$,

$$y_m[n] = -d_1 y_m[n-1] - d_2 y_m[n-2] - d_3 y_m[n-3] - d_4 y_m[n-4] + g_0 x[n] + g_1 x[n-1] + g_2 x[n-2].$$

In the identification phase, ten different sets of initial conditions (Run 1 ~ Run 10) are executed to validate the robustness of the proposed algorithm for each case. The best among these runs is taken to show the simulation results for the modeling and testing phases. Tab. 3 lists the cost functions derived by all of ten independent runs for Cases 1-3, respectively, where the optimal value appears in Run 8 for Case 1, Run 9 for Case 2, and Run 2 for Case 3. Based on these optimal independent runs of Cases 1-3, the comparisons of the actual and model outputs are shown in Figs. 6-14. Fig. 6 displays the model output of Run 8 with respect to the sampling number $0 \le n \le 100$ in the modeling phase for Case 1. It is clearly seen that both output curves are very close. Figs. 7 and 8 show the model outputs with respect to $0 \le n \le 200$ in the testing phase by the random and Gaussian testing inputs, respectively. Better approximations seen from these two figures are also derived. As to other simulation results, they are displayed in Figs. 9-14 for Cases 2 and 3, respectively. Again, the model output of each case is close to the actual output when both modeling and testing inputs are applied. Finally, numerical results derived by the proposed method are further compared with those obtained from a different method, and they are listed in Tab. 4. It can be concluded from Tab. 4 that the proposed method obtains better results over the other existing method.

5. Conclusions

In this paper, a modified Hammerstein model has been successfully applied to the nonlinear system modeling problem based on using differential evolution algorithm. The developed model is constructed using bilinear neural network followed by recursive digital system in order to provide better modeling capability. To adjust model parameters, the DE algorithm is utilized, so that the error signal between the actual and model outputs is minimized. Design steps of DE-based nonlinear system modeling are also described in detail. Finally, the developed method is applied to a model of a simulated chemical process, i.e. CSTR. Simulation results of modeling and testing phases are satisfactory and superior when compared to another existing method found in the subject literature.

References

[1] M. Cui, H. Liu, Z. Li, Y. Tang, X. Guan, Identification of Hammerstein model using functional link artificial neural network, Neurocomputing 142 (2014) 419-428.

[2] A. Gotmare, R. Patidar, N.V. George, Nonlinear system identification using a cuckoo search optimized adaptive Hammerstein model, Expert System with Applications 42 (2015) 2538-2546.

[3] M. Schoukens, A. Marconato, R. Pintelon, G. Vandersteen, Y. Rolain, Parametric identification of parallel Wiener-Hammerstein systems, Automatica 51 (2015) 111-122.

[4] J. Wang, Q. Zhang, Detection of asymmetric control valve stiction from oscillatory data using an extended Hammerstein system identification method, Journal of Process Control 24 (2014) 1-12.

[5] H. Chen, Y. Xiao, F. Ding, Hierarchical gradient parameter estimation algorithm for Hammerstein nonlinear systems using the key term separation, Applied Mathematics and Computation 247 (2014) 1202-1210.

[6] M. Rasouli, D. Westwick, W. Rosehart, Quasiconvexity analysis of the Hammerstein model, Automatica 50 (2014) 277-281.

[7] F. Yu, Z. Mao, M. Jia, P. Yuan, Recursive parameter identification of Hammerstein-Wiener systems with measurement noise, Signal Processing 105 (2014) 137-147.

[8] R. Storn, K. Price, Differential evolution — a simple and efficient heuristic for global optimization over continuous, Journal of Global Optimization 11 (1997) 341-359.

[9] S.P. Torres, C.A. Castro, Specialized differential evolution technique to solve

the alternating current model based transmission expansion planning problem, Electrical Power and Energy Systems 68 (2015) 243-251.

[10] S. Gundry, J. Zou, M.U. Uyar, C.S. Sahin, J. Kusyk, Differential evolution-based autonomous and disruption tolerant vehicular self-organization in MANETs, Ad Hoc Networks 25 (2015) 454-471.

[11] Y. Amrane, M. Boudour, A.A. Ladjici, A. Elmaouhab, Optimal VAR control for real power loss minimization using differential evolution algorithm, Electrical Power and Energy Systems 66 (2015) 262-271.

[12] A. Zamuda, J. Daniel, H. Sosa, Differential evolution and underwater glider path planning applied to the short-term opportunistic sampling of dynamic mesoscale ocean structure, Applied Soft Computing 24 (2014) 95-108.

[13] P. Kuila, P.K. Jana, A novel differential evolution based clustering algorithm for wireless sensor networks, Applied Soft Computing 25 (2015) 414-425.

[14] M. Basu, Improved differential evolution for economic dispatch, Electrical Power and Energy Systems 63 (2014) 855-861.

[15] R.D. Al-Dabbagh, A. Kinsheel, S. Mekhilef, M.S. Baba, S. Shamshirband, System identification and control of robot manipulation based on fuzzy adaptive differential evolution algorithm, Advances in Engineering Software 78 (2014) 60-66.

[16] Y. He, Q. Xu, S. Yang, A. Han, L. Yang, A novel chaotic differential evolution algorithm for short-term cascaded hydroelectric system scheduling, Electrical Power and Energy Systems 61 (2014) 455-462.

[17] A. Zamuda, J. Brest, Vectorized procedural models for animated trees reconstruction using differential evolution, Information Sciences 278 (2014) 1-21.

[18] I. Poikolainen, F. Neri, F. Caraffini, Cluster-based population initialization for differential evolution frameworks, Information Sciences 297 (2015) 216-235.

[19] A. Draa, S. Bouzoubia, I. Boukhalfa, A sinusoidal differential evolution algorithm for numerical optimisation, Applied Soft Computing 27 (2015) 99-126.

[20] Q. Fan, X. Yan, Self-adaptive differential evolution algorithm with discrete mutation control parameters, Expert System with Applications 42 (2015) 1551-1572.

[21] Y. Chen, W. Xie, X. Zou, A binary differential evolution algorithm learning from explored solutions, Neurocomputing 149 (2015) 1038-1047.

[22] X. Lu, K. Tang, B. Sendhoff, X. Yao, A new self-adaptation scheme for differential evolution, Neurocomputing 146 (2014) 2-16.

[23] W.D Chang, Recurrent neural network modeling combined with bilinear model structure, Neural Computing and Applications, DOI 10.1007/s00521-012-1295-5.

[24] C.T. Chen, S.T. Peng, Learning control of process systems with hard input constraints, Journal of Process Control 9 (1999) 151-160.

Figure captions

Table captions

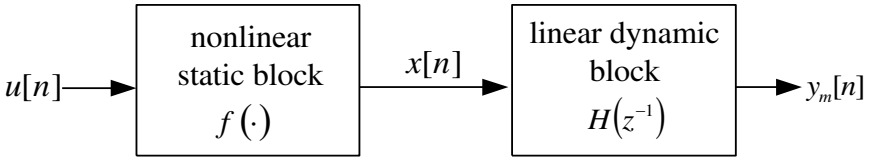

Fig. 1. A general Hammerstein model.

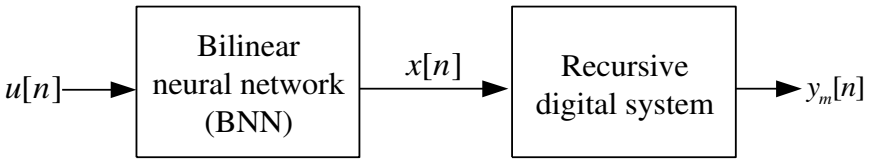

Fig. 2. A new modified Hammerstein model.

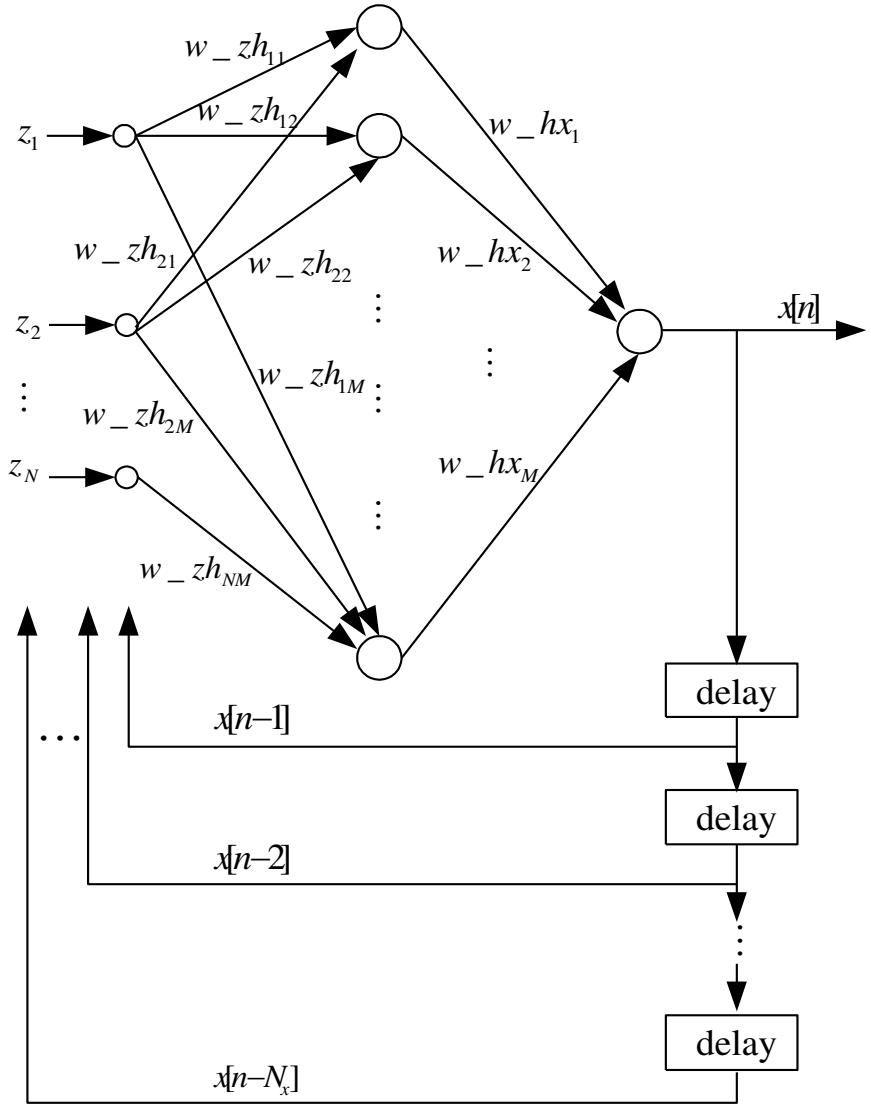

Fig. 3. BNN structure with N input neurons, M hidden neurons, and a single output neuron.

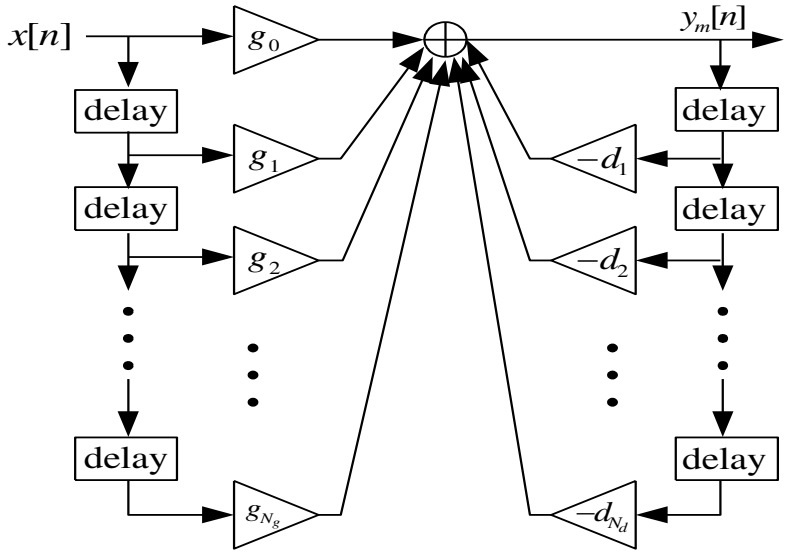

Fig. 4. Difference equation diagram for recursive digital system.

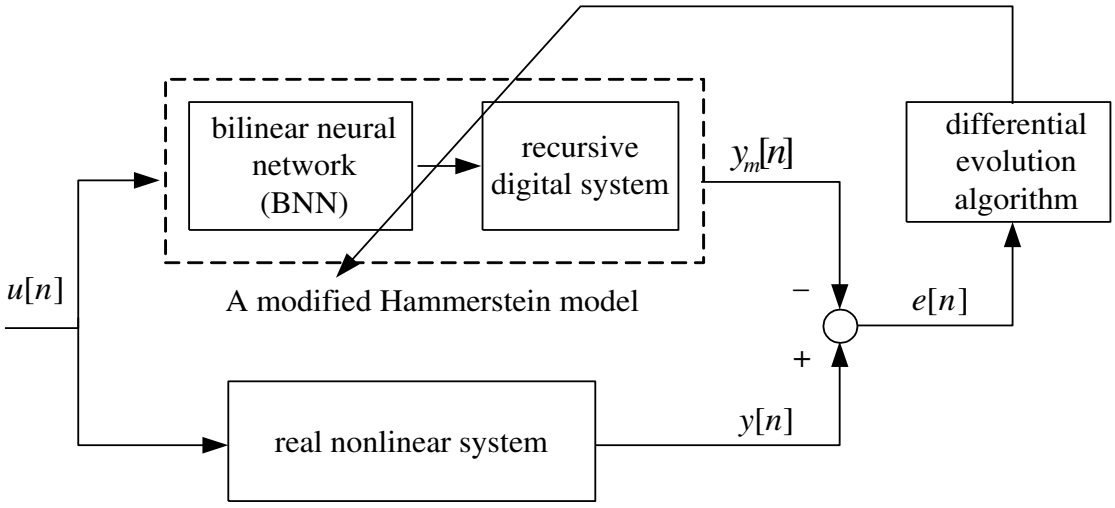

Fig. 5. The system block diagram of the proposed Hammerstein modeling.

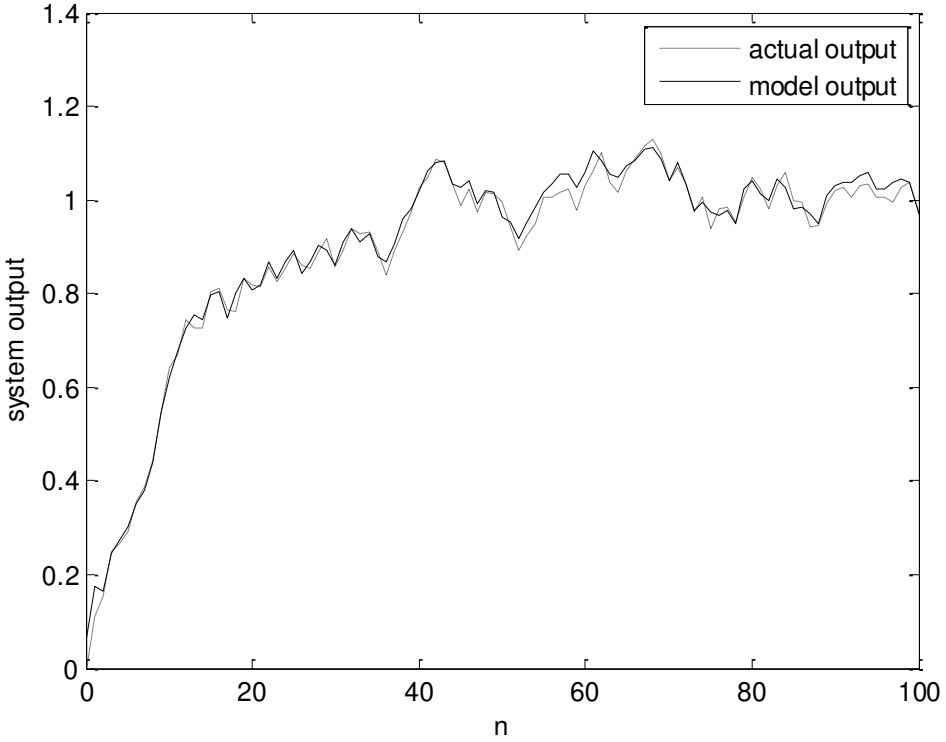

Fig. 6. Model output of Run 8 for Case 1 in the modeling phase.

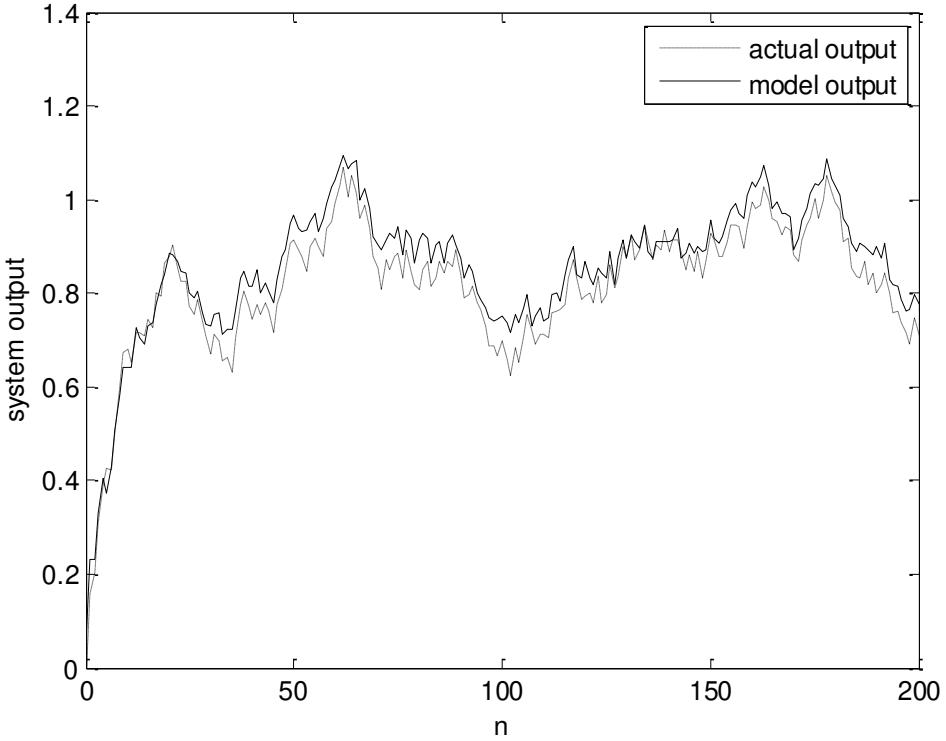

Fig. 7. Model output of Run 8 for Case 1 in the testing phase using the random input.

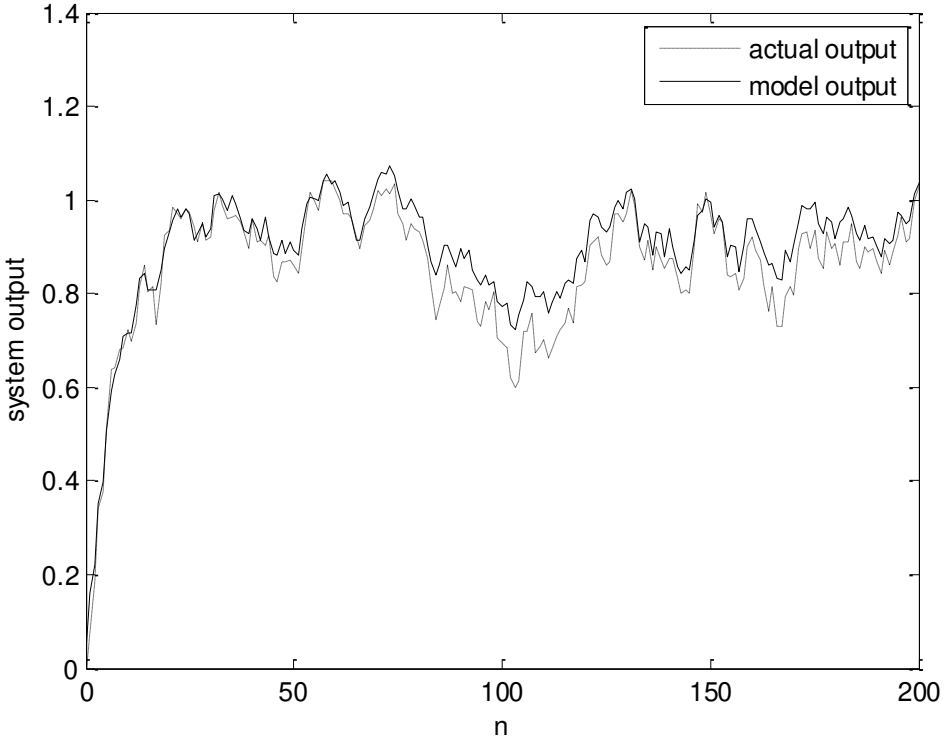

Fig. 8. Model output of Run 8 for Case 1 in the testing phase using the Gaussian input.

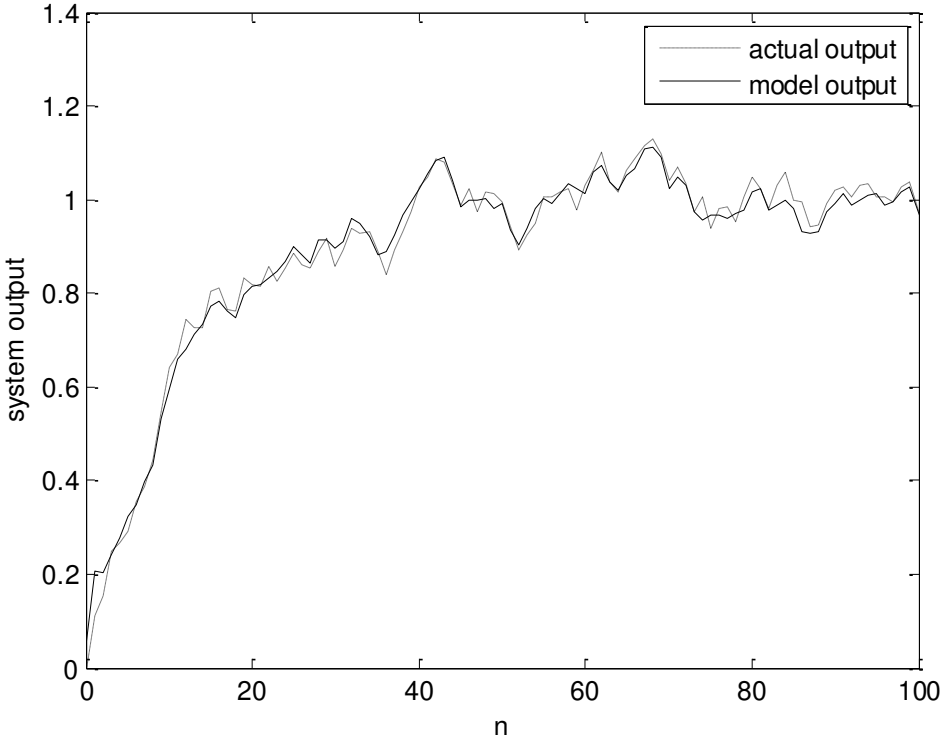

Fig. 9. Model output of Run 9 for Case 2 in the modeling phase.

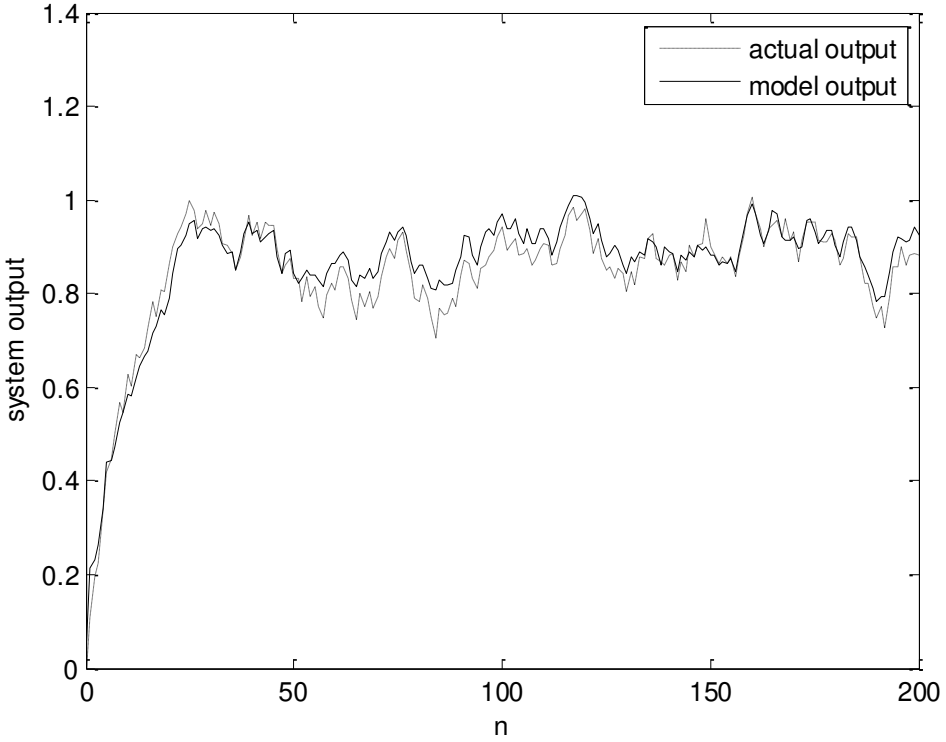

Fig. 10. Model output of Run 9 for Case 2 in the testing phase using the random input.

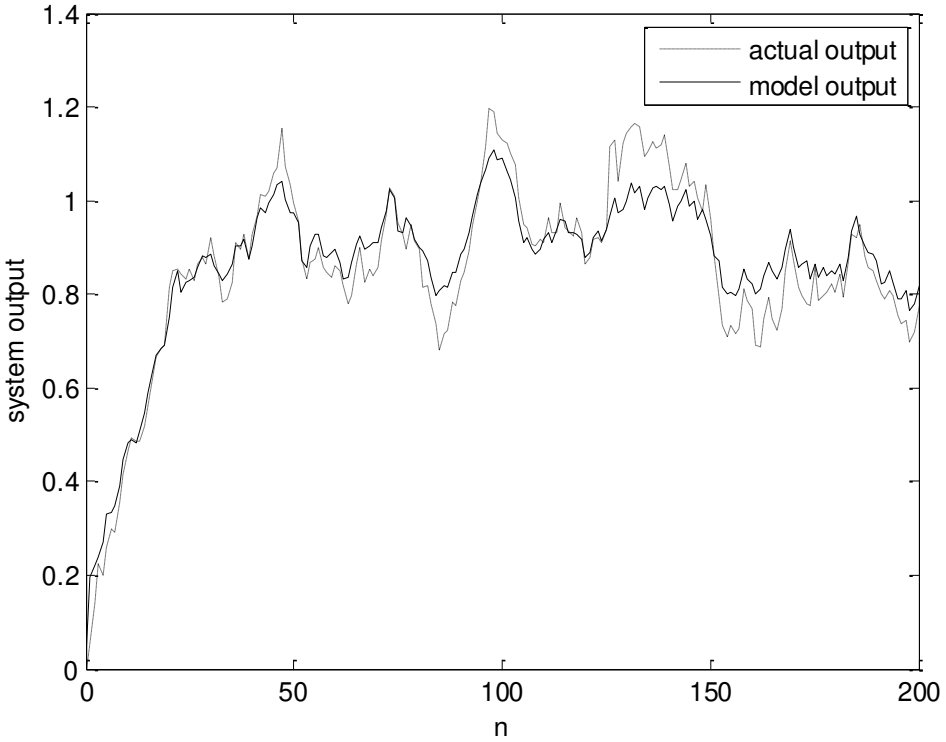

Fig. 11. Model output of Run 9 for Case 2 in the testing phase using the Gaussian input.

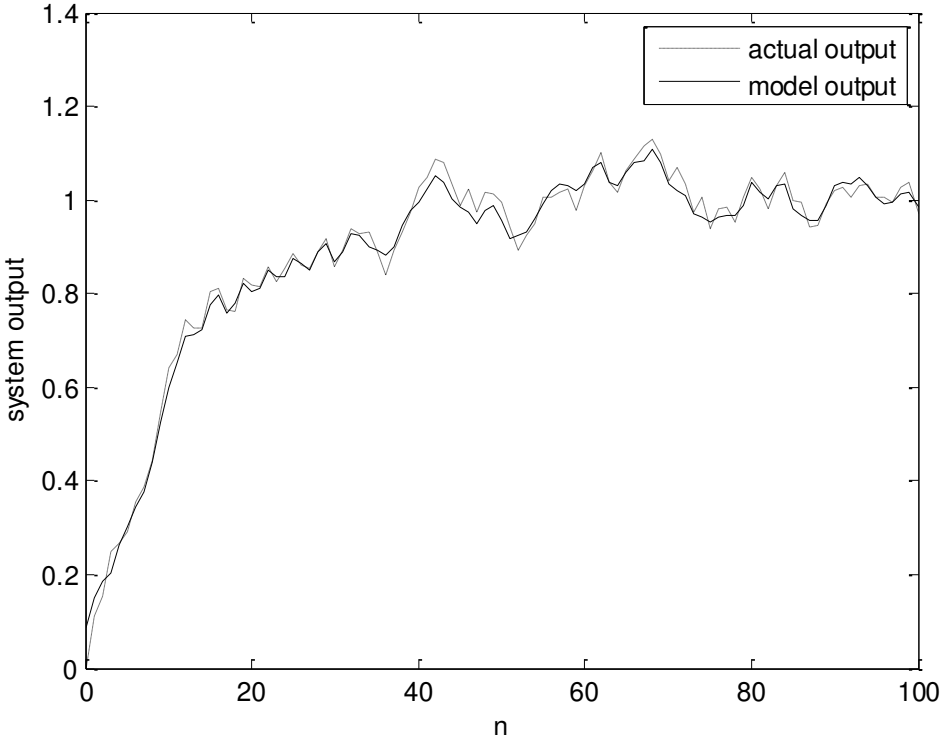

Fig. 12. Model output of Run 2 for Case 3 in the modeling phase.

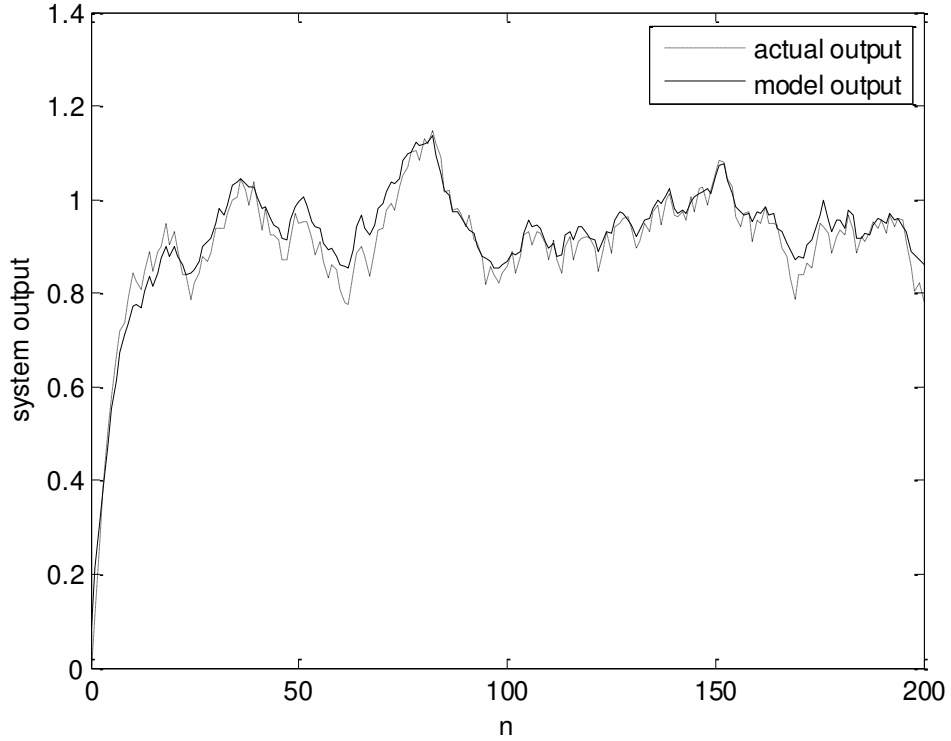

Fig. 13. Model output of Run 2 for Case 3 in the testing phase using the random input.

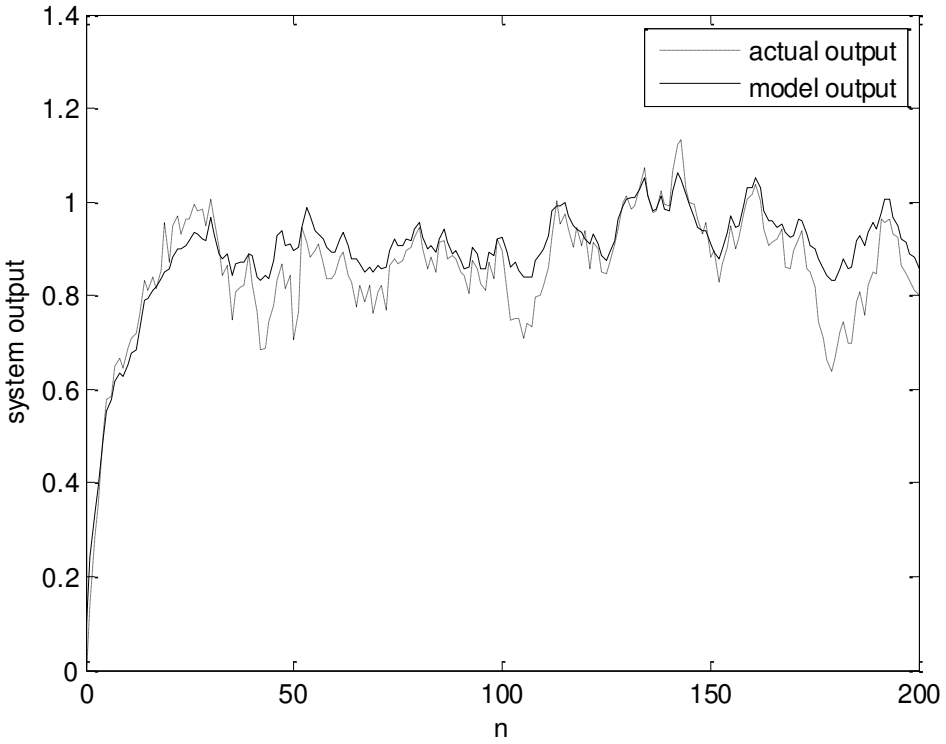

Fig. 14. Model output of Run 2 for Case 3 in the testing phase using the Gaussian input.

Tab. 1. Definition of CSTR system parameters.

Symbol	D_a	φ	B	δ
Definition	Damökhler number	activated energy	heat of reaction	heat transfer coefficient
Value	0.072	20	8	0.3

Tab. 2. Parameter setting used in the BNN and DE algorithm.

Orders of bilinear digital system	Number of hidden neurons	Population size	Mutation constant factor	Number of samples	Number of generations
$N_x = 5, N_u = 4$	$M = 5$	$PS = 30$	$f_s = 0.2$	$NS = 100$	$G\_num = 1000$

Tab. 3. Cost functions derived by ten independent runs for Cases 1-3.

	Case 1	Case 2	Case 3
Run 1	0.08283456	0.07306794	0.07526387
Run 2	0.14915122	0.07076396	**0.05209232**
Run 3	0.07813995	0.17962337	0.20358056
Run 4	0.08203442	0.11653909	0.05946590
Run 5	0.15542551	0.08635755	0.08316419
Run 6	0.06885503	0.11054810	0.06735277
Run 7	0.06991435	0.08199433	0.05924162
Run 8	**0.04398349**	0.09164785	0.14748451
Run 9	0.05463128	**0.06602896**	0.15223737
Run 10	0.05514363	0.21021702	0.09611215
Mean	0.084011344	0.108678818	0.099599526
Variance	0.001311015	0.002143345	0.002328653

Tab. 4. Numerical comparisons by the proposed and the other methods.

Used method		Mean	Variance
Proposed method	Case 1	0.084011344	0.001311015
	Case 2	0.108678818	0.002143345
	Case 3	0.099599526	0.002328653
Recurrent NN modeling [23]		0.344057590	0.014582315