Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3102845, 13 pages
http://dx.doi.org/10.1155/2016/3102845

Research Article

An Improved Global Harmony Search Algorithm for the Identification of Nonlinear Discrete-Time Systems Based on Volterra Filter Modeling

Zongyan Li[1] and Deliang Li[2]

[1]*School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu 221116, China*
[2]*Xuzhou College of Industrial Technology, Xuzhou, Jiangsu 221140, China*

Correspondence should be addressed to Zongyan Li; lizongyan@cumt.edu.cn

Received 6 November 2015; Revised 9 January 2016; Accepted 17 January 2016

Academic Editor: Erik Cuevas

This paper describes an improved global harmony search (IGHS) algorithm for identifying the nonlinear discrete-time systems based on second-order Volterra model. The IGHS is an improved version of the novel global harmony search (NGHS) algorithm, and it makes two significant improvements on the NGHS. First, the genetic mutation operation is modified by combining normal distribution and Cauchy distribution, which enables the IGHS to fully explore and exploit the solution space. Second, an opposition-based learning (OBL) is introduced and modified to improve the quality of harmony vectors. The IGHS algorithm is implemented on two numerical examples, and they are nonlinear discrete-time rational system and the real heat exchanger, respectively. The results of the IGHS are compared with those of the other three methods, and it has been verified to be more effective than the other three methods on solving the above two problems with different input signals and system memory sizes.

1. Introduction

The Volterra model is a kind of nonlinear filter model, which is usually employed to track and identify plenty of complex nonlinear systems. In order to enhance the quality of the system identification, it is very crucial issue to select optimum model coefficient called the kernel. Therefore, the Volterra model is essentially an extension of linear filter model to nonlinear case. During the past decade, there have been many research works on the Volterra model. Campello et al. [1] tackled the problem of expanding Volterra models using Laguerre functions. The global optimal solution is obtained when each multidimensional kernel of the model is decomposed into a number of independent orthonormal bases. Furthermore, the solution obtained is able to minimize the upper bound of the squared norm of the error resulting from the practical truncation of the Laguerre series expansion into a finite number of functions. Masugi and Takuma [2] described a Volterra system-based nonlinear study of video-packet transmission over IP networks. Based on the Volterra system, the authors performed a time-series analysis of measured data for network response evaluation. The novel method can reproduce the time-series responses observed in video-packet transmission over the Internet, characterizing nonlinear dynamic behaviors such that the obtained results gave an appropriate depiction of network conditions at different times. Gruber et al. [3] presented a nonlinear model predictive control (NMPC) method based on a second-order Volterra series model for greenhouse temperature control using natural ventilation. These models, denoting the simple and logical extension of convolution models, are capable of describing the nonlinear dynamic feature of the ventilation and other environmental conditions on the greenhouse temperature. Many applications of Volterra series modeling were executed in the frequency-domain based on the Generalized Frequency Response Functions (GFRF) [4, 5]. In the light of the wide applications of Volterra series, Li and Billings [6] presented an approach to estimate the GFRF in a piecewise manner for duffing type oscillators for the underrepresented weakly nonlinear region. Additionally,

they devised a new method to obtain the energy contributions of each order kernel. The new nonparametric method can not only construct the amplitude-invariant GFRF over a certain excitation range, but also avoid building large sets of time-domain models. In addition, the Volterra filter model can be found in some other application areas [7–14].

Chang [7] devised an improved particle swarm optimization (IPSO) to implement system identification based on Volterra filter model. In this paper, we develop an improved global harmony search (IGHS) algorithm and try the IGHS as an efficient candidate for system identification based on Volterra filter model. The harmony search (HS) algorithm was firstly proposed by [15]. The HS is a simple but efficient algorithm, and its many improved versions have been applied into many problems including reliability problems [16], reactor core fuel management optimization [17], and sizing optimization of truss structures [18].

The paper is organized as follows. In Section 2, the pruned second-order Volterra filter model is simply presented. In Section 3, a novel global harmony search algorithm is introduced. In Section 4, an improved global harmony search algorithm is proposed, and its procedure is fully explained. In Section 5, four harmony search algorithms are used for two examples with different signal inputs and memory sizes. We end this paper with some conclusions in Section 6.

2. Volterra Filter Model and Its Pruned Second-Order Form

The Volterra filter model is an efficient method for the identification of nonlinear discrete systems, and it has come into researchers' notice in recent decades. The discrete form of Volterra filter model of the qth order [7] is given by

$$\begin{aligned} y[n] &= h_0 + \sum_{k_1=0}^{N-1} h_1[k_1]\, x[n-k_1] \\ &\quad + \sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1} h_2[k_1,k_2]\, x[n-k_1]\, x[n-k_2] + \cdots \\ &\quad + \sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1}\cdots\sum_{k_q=0}^{N-1} h_q[k_1,k_2,\ldots,k_q] \prod_{i=1}^{q} x[n-k_i] \\ &= h_0 \\ &\quad + \sum_{j=1}^{q}\sum_{k_1=0}^{N-1}\cdots\sum_{k_j=0}^{N-1} h_j[k_1,k_2,\ldots,k_j] \prod_{i=1}^{j} x[n-k_i], \end{aligned} \tag{1}$$

where N denotes the system memory size. Equation (1) denotes the Volterra filter model with the infinite series. However, this model is hard to compute and master due to its complex and expatiatory formula. In this paper, we only study its simplified and approximate form called the truncated second-order Volterra model [7, 19], which is stated as follows:

$$\begin{aligned} y[n] &= h_0 + \sum_{k=1}^{N} h[k]\, x[n-k+1] \\ &\quad + \sum_{k_1=0}^{N-1}\sum_{k_2=k_1}^{N-1} h[k_1,k_2]\, x[n-k_1]\, x[n-k_2]. \end{aligned} \tag{2}$$

In order to facilitate expression, (2) can also be expressed as the following vector form:

$$y[n] = HX^T. \tag{3}$$

Here, the superscript T represents the transpose of a vector and H stands for the Volterra kernel vector given by

$$\begin{aligned} H = [&h_0, h[1], \ldots, h[N], h[0,0], h[0,1], \ldots, h[0,N-1], \\ &h[1,1], \ldots, h[N-1,N-1]]. \end{aligned} \tag{4}$$

In addition, X denotes the Volterra input vector given by

$$\begin{aligned} X = \big[&1, x[n], \ldots, x[n-N+1], x^2[n], x[n] \\ &\cdot x[n-1], \ldots, x[n] \\ &\cdot x[n-(N-1)], x^2[n-1], \ldots, x^2[n-(N-1)]\big]. \end{aligned} \tag{5}$$

In the light of (3), the vector lengths of both H and X are the same and are calculated as follows [19]:

$$L = C_{N+2}^{N} = 1 + N + \frac{N(N+1)}{2} = \frac{(N+1)(N+2)}{2}. \tag{6}$$

To achieve the nearest approximation of the actual system output, appropriate kernel vector H should be determined under the input vector X. In this paper, an improved global harmony search (IGHS) algorithm is proposed to determine kernel vector H. The IGHS is an improved version of novel global harmony search (NGHS) algorithm [20]; thus, both the NGHS and the IGHS will be presented in the following sections.

3. Novel Global Harmony Search (NGHS) Algorithm

Novel global harmony search (NGHS) algorithm [20] is a variant of harmony search (HS) algorithm [15], and it is superior to the HS for solving unconstrained optimization problems. The NGHS improvises new harmony vectors by combining position updating and mutation. Concretely, the steps of NGHS are explained as follows.

Step 1 (initialize the NGHS parameters and the problem parameters). The NGHS parameters consist of harmony memory size HMS, the number of improvisations NI, and mutation rate p_m. In addition, the problem parameters include the number of problem variables L, the lower bound $\underline{x}_j$, and upper bound $\overline{x}_j$ of the jth ($j = 1, 2, \ldots, L$) problem

variable. Furthermore, the number of improvisations (NI) is actually the total number of generations for adjusting the parameters related to the identification problem. In each generation, only one new candidate solution including L parameters is generated, and this solution is accepted if and only if it is better than the worst one of the previous solutions.

Step 2 (initialize harmony memory (HM)). The initial harmony memory (HM) can be expressed in the following matrix form:

$$\mathrm{HM} = \begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_L^1 \\ x_1^2 & x_2^2 & \cdots & x_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_L^{\mathrm{HMS}} \end{pmatrix}, \tag{7}$$

where x_j^i stands for the jth ($j = 1, \ldots, L$) variable of the ith ($i = 1, \ldots, \mathrm{HMS}$) harmony vector. Moreover, it is randomly produced from a uniform distribution in the ranges $[\underline{x}_j, \overline{x}_j]$ ($j = 1, 2, \ldots, L$). In HM, any vector $x^i = (x_1^i, x_2^i, \ldots, x_L^i)$ ($i = 1, \ldots, \mathrm{HMS}$) represents a candidate solution of the parameters (as in (4)) needed for solving the identification problem. Furthermore, the length of x^i is equal to L, which is exactly the number of the parameters in the Volterra kernel vector H (as in (6)).

Step 3 (improvise a new harmony). Improvisation is actually the operation of producing a new harmony vector. For the NGHS, its improvisation mainly includes two steps, and they are position updating and genetic mutation, respectively. More specifically, the improvisation can be presented in Table 1.

Here, "best" and "worst" stand for the indexes of the best harmony and the worst harmony in HM, respectively. r_1, r_2, and r_3 denote three uniformly generated random numbers in $[0, 1]$. With respect to position updating, a new harmony vector is improvised near the best harmony vector, which can facilitate the convergence rate of the NGHS. On the other hand, it is worth noticing that genetic mutation is an event of small probability, and it is utilized to avoid the premature convergence of the NGHS.

Step 4 (update harmony memory). Replace the worst harmony vector x^{worst} of HM with the new improvised harmony vector x' no matter whether x' is better or worse than x^{worst}.

Step 5 (check the stopping criterion). Steps 3 and 4 are repeated until the number of improvisations (NI) is reached.

4. An Improved Global Harmony Search (IGHS) Algorithm

In this paper, an improved global harmony search (IGHS) algorithm is proposed to implement the system identification based on Volterra filter model. The IGHS is an improved version of the NGHS, and it is different from the NGHS in the following two aspects.

Table 1: The computational procedure of the NGHS improvisation.

Line	The improvisation of the NGHS
(1)	**For** $j = 1$ to L
(2)	$x_R = 2 \times x_j^{\mathrm{best}} - x_j^{\mathrm{worst}}$
(3)	**If** $x_R > \overline{x}_j$
(4)	$x_R = \overline{x}_j$
(5)	**ElseIf** $x_R < \underline{x}_j$
(6)	$x_R = \underline{x}_j$
(7)	**EndIf**
(8)	$x_j' = x_j^{\mathrm{worst}} + r_1 \times (x_R - x_j^{\mathrm{worst}})$% Position updating
(9)	**If** $r_2 \le p_m$
(10)	$x_j' = \underline{x}_j + r_3 \times (\overline{x}_j - \underline{x}_j)$% Genetic mutation
(11)	**EndIf**
(12)	**EndFor**

(1) The Modification of Genetic Mutation. The genetic mutation of the NGHS is conducted according to uniform distribution. To further explore and exploit the solution space, the uniform distribution is replaced with the other two probability distributions, and they are normal distribution [21–23] and Cauchy distribution [22, 23], respectively. Therefore, the new genetic mutation are implemented by using the following two equations:

$$\begin{aligned} x_j' &= \mathrm{rand}\, n_j\left(x_j^{\mathrm{best}}, 0.1\right), \\ x_j' &= \mathrm{rand}\, c_j\left(x_j^{\mathrm{best}}, 0.1\right). \end{aligned} \tag{8}$$

Here, j denotes the index of the jth component of the improvised harmony vector, and x_j^{best} denotes the jth component of the best harmony vector in HM. $\mathrm{rand}\, n_j(x_j^{\mathrm{best}}, 0.1)$ stands for the normal distribution with mean x_j^{best} and standard deviation 0.1, and $\mathrm{rand}\, c_j(x_j^{\mathrm{best}}, 0.1)$ stands for the Cauchy distribution with location parameter x_j^{best} and scale parameter 0.1. If overflow happens to x_j' generated using either normal distribution or Cauchy distribution, it will be truncated to $[\overline{x}_j, \underline{x}_j]$. On the other hand, if the condition $r_2 < p_m$ (as in Table 1) is satisfied, either normal distribution or Cauchy distribution will be utilized to carry out genetic mutation, and the probability of using any one of the two distributions is equal to 0.5. By adopting normal distribution and Cauchy distribution, the capacity of escaping from the local optimums is enhanced for the IGHS. In the meantime, the solution space can be fully explored and exploited, which is beneficial for improving the quality of the improvised harmony vector x'.

In addition to the equation of genetic mutation, we also modify the mutation rate (p_m) of the NGHS. The parameter p_m is used to determine whether or not a variable of x' adopts genetic mutation. A large p_m value is beneficial for searching in a large scope, while a small p_m value is helpful to search

in a small scope. To well balance the global search and local search of the IGHS, the p_m value decreases dynamically with increasing generations (NI) as follows:

$$p_m^t = \begin{cases} \overline{p}_m - \dfrac{\overline{p}_m - \underline{p}_m}{\mathrm{NI}_0} \times t, & \text{if } t \le \mathrm{NI}_0; \\ \underline{p}_m, & \text{if } t > \mathrm{NI}_0, \end{cases} \tag{9}$$

where $\underline{p}_m$ and $\overline{p}_m$ represent the minimal and maximal mutation rates. t ($1 \le t \le \mathrm{NI}$) denotes the current generation number, and the IGHS is stopped when the current generation number t reaches the maximal generation number NI. In addition, NI_0 denotes a fixed generation number, and it is set to 3NI/4 here.

(2) Introduce and Modify an Opposition-Based Learning (OBL) [24]. In order to improve the convergence of the IGHS, a method called opposition-based learning (OBL) [24] is firstly introduced. In short, this technique can be stated as follows:

$$ox_j^i = x_j^L + x_j^U - x_j^i. \tag{10}$$

Here, ox^i is the opposite solution of x^i, and ox_j^i is the jth variable of ox^i. Furthermore, x_j^L and x_j^U represent the minimal and the maximal values in the set $\{x_j^1, x_j^2, \ldots, x_j^{\mathrm{HMS}}\}$, and they are expressed as follows:

$$\begin{aligned} x_j^L &= \min_{1 \le i \le \mathrm{HMS}} \left(x_j^i\right), \\ x_j^U &= \max_{1 \le i \le \mathrm{HMS}} \left(x_j^i\right). \end{aligned} \tag{11}$$

According to (10), only the opposite value of x_j^i is employed to produce the corresponding variable in the range of $[x_j^L, x_j^U]$. In other words, once the x_j^i value is determined, it has an exclusive opposite value ox_j^i which is equal to $x_j^L + x_j^U - x_j^i$.

In this paper, we enhance the IGHS improvisation by modifying the OBL technique. More specifically, the ox_j^i value is randomly produced in the range of $[x_j^L, x_j^U]$, and it is given by

$$ox_j^i = x_j^L + \mathrm{rand}\,() \times \left(x_j^U - x_j^L\right), \tag{12}$$

where rand() denotes a random number uniformly generated in [0, 1]. By using (12), the IGHS can yield numerous possible values in the range of $[x_j^L, x_j^U]$. Compared to the OBL technique, its modified version has wider searching space, which is in favor of fine-tuning of the best harmony vectors.

Based on (10)-(11), a new harmony memory HM' can be generated, and it is given by

$$\mathrm{HM}' = \begin{pmatrix} ox_1^1 & ox_2^1 & \cdots & ox_L^1 \\ ox_1^2 & ox_2^2 & \cdots & ox_L^2 \\ \vdots & \vdots & \ddots & \vdots \\ ox_1^{\mathrm{HMS}} & ox_2^{\mathrm{HMS}} & \cdots & ox_L^{\mathrm{HMS}} \end{pmatrix}. \tag{13}$$

Table 2: The computational procedure of the IGHS improvisation.

Line	The improvisation of the IGHS
(1)	**For** $j = 1$ to L
(2)	$x_R = 2 \times x_j^{\mathrm{best}} - x_j^{\mathrm{worst}}$
(3)	**If** $x_R > \overline{x}_j$
(4)	$x_R = \overline{x}_j$
(5)	**ElseIf** $x_R < \underline{x}_j$
(6)	$x_R = \underline{x}_j$
(7)	**EndIf**
(8)	$x_j' = x_j^{\mathrm{worst}} + r_1 \times (x_R - x_j^{\mathrm{worst}})$% Position updating
(9)	**If** $r_2 \le p_m^t$% p_m^t is calculated according to (9)
(10)	**If** $r_3 \le 0.5$
(11)	$x_j' = \mathrm{rand}n_j(x_j^{\mathrm{best}}, 0.1)$% Normal distribution
(12)	**Else**
(13)	$x_j' = \mathrm{rand}c_j(x_j^{\mathrm{best}}, 0.1)$% Cauchy distribution
(14)	**EndIf**
(15)	**If** $x_j' > \overline{x}_j$
(16)	$x_j' = \overline{x}_j$
(17)	**ElseIf** $x_j' < \underline{x}_j$
(18)	$x_j' = \underline{x}_j$
(19)	**EndIf**
(20)	**EndIf**
(21)	**EndFor**
(22)	**If** $r_4 < 0.1$% Modified OBL operation with a probability of 0.1
(23)	**For** $i = 1$ to HMS
(24)	**For** $j = 1$ to L
(25)	$ox_j^i = x_j^L + \mathrm{rand}() \times (x_j^U - x_j^L)$
(26)	**EndFor**
(27)	**If** $f(ox^i) < f(x^i)$
(28)	$x^i = ox^i$
(29)	**Endif**
(30)	**EndFor**
(31)	**EndIf**

Here, $ox^i = (ox_1^i, ox_2^i, \ldots, ox_L^i)$ stands for the ith ($i = 1, \ldots, \mathrm{HMS}$) harmony vector in HM', and ox_j^i denotes its jth ($j = 1, \ldots, L$) variable. After performing the modified OBL operation, each ox^i is compared with the corresponding harmony vector x^i. More precisely, if ox^i is better than x^i, x^i should be replaced with ox^i. It is worth noticing that the modified OBL is an event of small probability. In other words, this technique is an auxiliary step of the IGHS improvisation, and it is mainly used to improve the quality of the improvised harmony vector.

Based on the above detailed illustrations about the two modifications of the NGHS improvisation, the IGHS improvisation can be summarized in Table 2.

By utilizing the IGHS, the identification diagram of nonlinear discrete-time systems based on the pruned second-order Volterra model is shown in Figure 1. Besides, the nomenclatures appearing in this diagram are explained in Table 3.

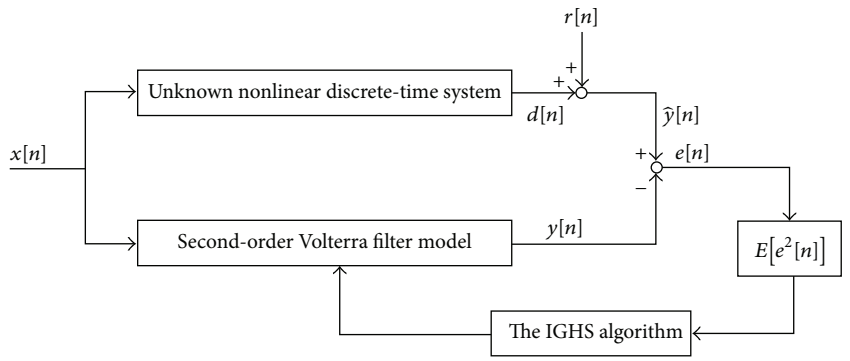


Figure 1: Volterra filter modeling of nonlinear discrete-time system using the IGHS algorithm.

Table 3: Nomenclatures used in Volterra filter modeling of nonlinear discrete-time system.

$x[n]$	The digital input signal
$y[n]$	The output signal of the second-order Volterra filter model (as (2))
$d[n]$	The output of the unknown nonlinear discrete-time system
$r[n]$	The measurement noise
$\hat{y}[n]$	The sum of $d[n]$ and $r[n]$
$e[n]$	The error signal between $\hat{y}[n]$ and $y[n]$

The goal of the IGHS is to find the optimal kernel vector H for Volterra filter model so that the difference between the estimated output $y[n]$ and the actual system output $\hat{y}[n]$ is minimized. Thus, it is advisable to find an objective function to satisfy the design requirement. Here we adopt the following objective function [7]:

$$E\left[e^2[n]\right] = \frac{1}{T}\sum_{n=0}^{T-1} e^2[n] = \frac{1}{T}\sum_{n=0}^{T-1}\left[\hat{y}[n] - y[n]\right]^2. \quad (14)$$

Here, $E[e^2[n]]$ stands for the mean square error (MSE), and T denotes the sampling number. By using the IGHS to optimize MSE, we can obtain the most appropriate kernel vector for the second-order Volterra model.

5. Experimental Results and Analysis

To verify the validity of the IGHS on identifying nonlinear system based on second-order Volterra filter model, two examples including the highly nonlinear discrete-time rational system and the real heat exchanger are considered. More specifically, these two examples are explained as follows.

5.1. Example 1. The first example is the highly nonlinear discrete-time rational system [7], and its mathematical model is stated as follows:

$$d[n] = \frac{0.3d^2[n-1] + 0.8x[n-1] + 0.6d[n-2]}{1 + x^2[n-1] + d^2[n-1]}. \quad (15)$$

Here, two types of input signals [7] are considered. The first is defined as Example 1a whose input is a random signal uniformly produced in $[-1, 1]$. The second is defined as Example 1b whose input is a cosine signal $x[n] = 0.8\cos((\pi/9)n)$. For both Examples 1a and 1b, the measurement noise $r[n]$ is always supposed to be a Gaussian noise of $N(0, 0.001)$.

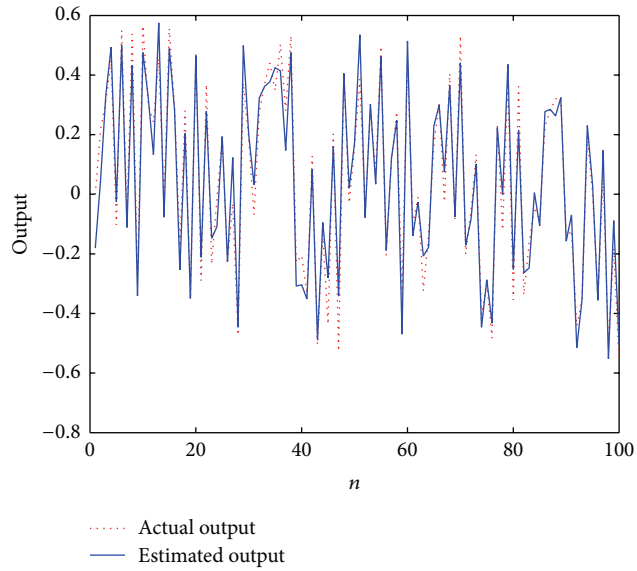


Figure 2: Comparison of actual system output and Volterra model output for Example 1a ($N = 5$).

In this experiment, the IGHS is used for Example 1a with $N = 5$. Moreover, the IGHS parameters are set as follows: the maximal mutation rate $\overline{p}_m = 0.1$, the minimal mutation rate $\underline{p}_m = 0.1$, harmony memory size HMS $= 5$, the number of improvisations NI $= 2000$, and sampling number $T = 100$. Moreover, T is actually the total number of output samples (actual system output or estimated output). According to (14), there are T sampling values for actual system output $\hat{y}[n]$ $(1 \le n \le T)$. After implementing the IGHS, we can obtain the comparison between the estimated output $y[n]$ and the actual output $\hat{y}[n]$ in Figure 2.

From Figure 2, it can be seen that the estimated output $y[n]$ approximates the actual output $\hat{y}[n]$ well. In addition, the minimal mean square error (MSE) obtained by the IGHS

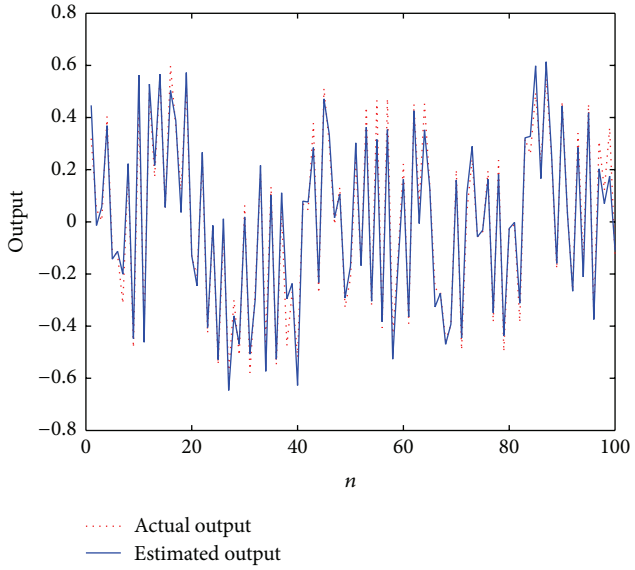


Figure 3: Comparison of actual system output and Volterra model output for Example 1a ($N = 8$).

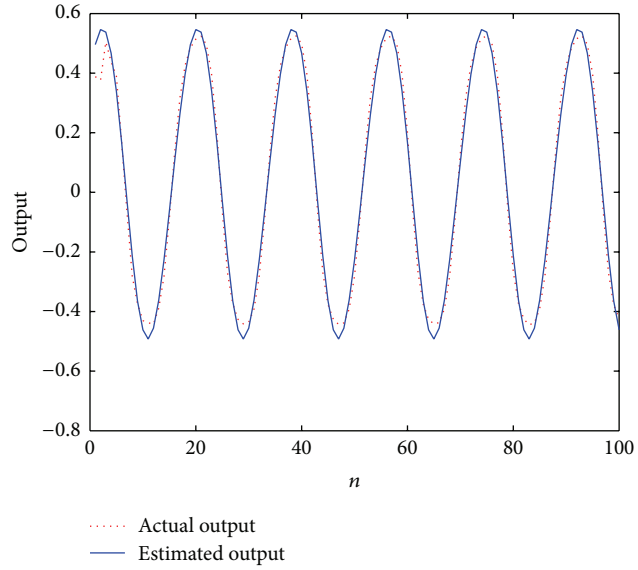


Figure 4: Comparison of actual system output and Volterra model output for Example 1b ($N = 5$).

is equal to 5.4594e−003, which is a satisfactory result. Besides, we use Example 1a with $N = 8$ to investigate the performance of the IGHS for second-order Volterra filter model with large memory size. The parameters of the IGHS are the same as those used for Example 1a ($N = 5$) except NI, and it is set to 5000 in this experiment. By performing the IGHS, we can obtain the comparison between the estimated output $y[n]$ and the actual output $\widehat{y}[n]$ in Figure 3.

It is clear from Figure 3 that a satisfactory approximation can be attained by utilizing the IGHS. Additionally, the minimal MSE yielded by the IGHS is equal to 3.7074e − 003 for Example 1a with $N = 8$, which provides better modeling capacity.

In addition to random signal, another testing input signal $x[n] = 0.8\cos((\pi/9)n)$ is used to investigate the performance of second-order Volterra filter model using the IGHS. For Example 1b with $N = 5$ and $N = 8$, the IGHS parameters are the same as those for Example 1a, and Figures 4 and 5 display the comparisons of results for Example 1b with $N = 5$ and $N = 8$. As expected, the difference between the estimated output $y[n]$ and the actual output $\widehat{y}[n]$ is small in each case. Additionally, two satisfying MSEs can be obtained, respectively, for $N = 5$ and $N = 8$ by carrying out the IGHS, and they are equal to 1.71809876e − 003 and 1.7119e − 003, respectively.

5.2. Example 2. The second example is the real heat exchanger and its mathematical model [7, 25] is given by

$$w[n] = x[n] - 1.3228x^2[n] + 0.7671x^3[n] - 2.1755x^4[n], \quad (16)$$

$$d[n] = \frac{-6.5306z^{-1} + 5.5652z^{-2}}{1 - 1.608z^{-1} + 0.6385z^{-2}} w[n], \quad (17)$$

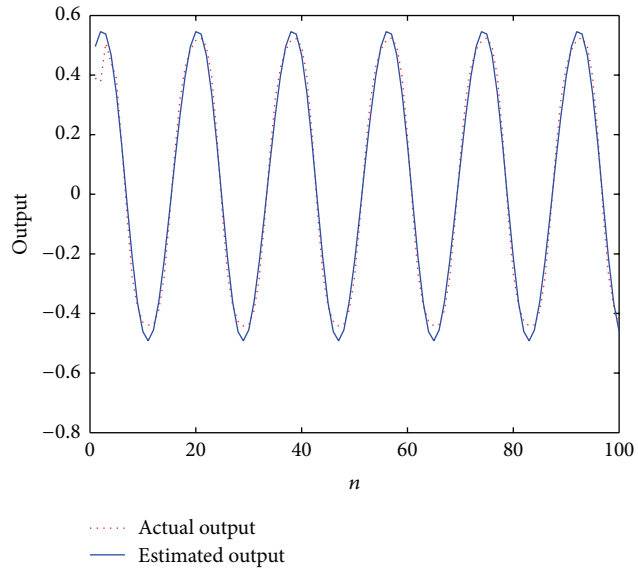


Figure 5: Comparison of actual system output and Volterra model output for Example 1b ($N = 8$).

where $x[n]$ is the process input denoting the flow rate and is constrained by the range of $[0, 1]$, $w[n]$ stands for the static nonlinearity, and $d[n]$ stands for the process output temperature. By combining (16) and (17), the simplified mathematical model can be expressed as the following difference equation:

$$d[n] = 1.608d[n-1] - 0.6385d[n-2] - 6.5306w[n-1] + 5.5652w[n-2]. \quad (18)$$

Here, two types of input signals [7, 25] are considered. The first is defined as Example 2a whose input is randomly generated in the range $[0.1, 0.9]$. The second is defined as Example

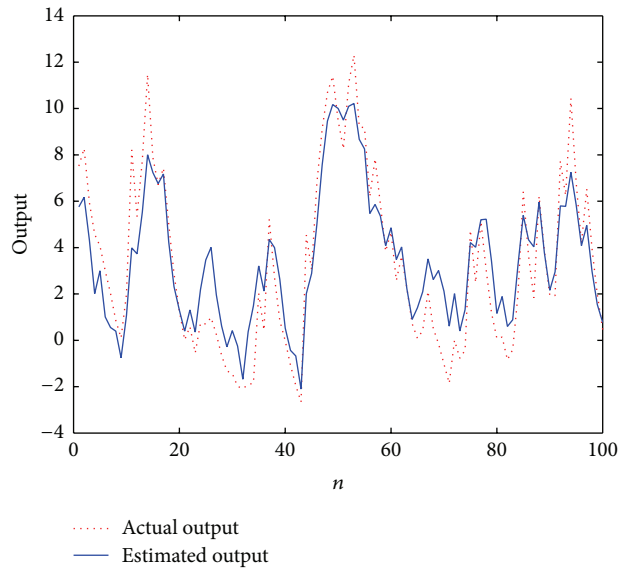


Figure 6: Comparison of actual system output and Volterra model output for Example 2a ($N = 8$).

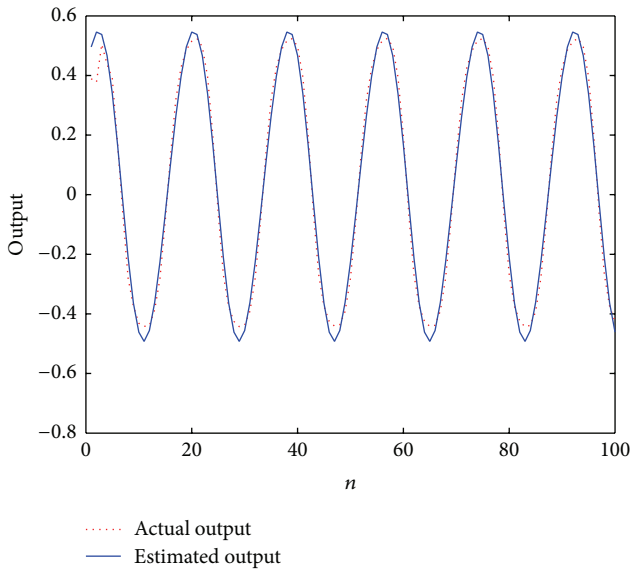


Figure 7: Comparison of actual system output and Volterra model output for Example 2b ($N = 8$).

2b whose input is a sine signal $x[n] = 0.4\sin((\pi/6)n) + 0.5$. According to [7, 25], the measurement noise is excluded from the problem model; thus, we have $r[n] = 0$.

In this experiment, the memory size N is set to 8 for second-order Volterra filter model. To fulfill the physical input requirement, the modeling input $x[n]$ is randomly generated in the range $[0.1, 0.9]$ and the testing input is then set to $x[n] = 0.4\sin((\pi/6)n) + 0.5$. Moreover, the IGHS parameters used for Example 2 are the same as those of Example 1. By implementing the IGHS, the outputs of second-order Volterra model are shown in Figures 6 and 7, respectively, for the modeling input and testing input. Based on the careful observations on experimental results, we can confirm that the satisfactory approximation is attained in each case.

5.3. Comparison of the IGHS with the Other Three HSs. In this paper, four methods consisting of the HS [15], IHS [26], NGHS [20], and IGHS are used to solve the above problems with different input signals and system memory sizes. The parameters of these four methods are set as follows. For the HS, harmony memory considering rate HMCR = 0.95, pitch adjusting rate PAR = 0.3, and bandwidth bw = 0.01. For the IHS, HMCR = 0.95, the minimal pitch adjusting rate $\mathrm{PAR}_{\min} = 0.35$, the maximal pitch adjusting rate $\mathrm{PAR}_{\max} = 0.99$, the minimal bandwidth $\mathrm{bw}_{\min} = 10^{-5}$, and the maximal bandwidth $\mathrm{bw}_{\max} = 0.1$. For the NGHS, mutation rate $p_m = 5 \times 10^{-2}$. For the IGHS, the minimal mutation rate $\underline{p}_m = 0.01$, the maximal mutation rate $\overline{p}_m = 0.1$, and $\mathrm{NI}_0 = 3\mathrm{NI}/4$. Note that all the methods are executed under identical conditions (harmony memory size and the number of improvisations). More specifically, the harmony memory size (HMS) of each method is set to 5, and the NI (Number of improvisations) value of each method is set to 2000 and 5000, respectively, for the system memory sizes $N = 5$ and $N = 8$. With respect to second-order Volterra filter model, sampling number is set to $T = 100$, and the range of each variable of the kernel vector is set as $[-1, 1]$. Matlab 7.0 is used to execute the above procedure under the environment of Intel(R) Core(TM) i5-2410M CPU @ 2.30 GHz. 20 independent runs are performed in each case, and the optimization results are presented in Table 4.

Table 4 gives the comparison of the results obtained by the IGHS, against the other three methods including the HS [15], the IHS [26], and the NGHS [20], and the best performance is reported in boldface. The terms "ACT" and "Std" stand for average computation time and standard deviation, respectively. According to the results, we can see that the IGHS performs better most of the time. To be precise, the values of Worst, Mean, and Std obtained by the IGHS are smaller than those obtained by the other three methods for all problems. Moreover, the IGHS can obtain five best objective function values except Example 1b ($N = 8$). Therefore, the IGHS has exhibited stronger convergence and stability than the other three HSs. For Example 1b ($N = 8$), the IHS, NGHS, and IGHS can yield the same best objective function value which is equal to $1.7119e - 003$, and the remaining results obtained by the IGHS are better than the other three HSs.

Additionally, Mann-Whitney U test [27, 28], also known as "Mann-Whitney Wilcoxon test," is used to ensure a statistical significant difference between the IGHS and any of the other three HSs. In other words, the Mann-Whitney U test acts as an efficient nonparametric rank-based test to identify a difference between populations. Moreover, the test statistic U is expressed as follows:

$$U_1 = R_1 - \frac{n_1(n_1 + 1)}{2}, \tag{19}$$

$$U_2 = R_2 - \frac{n_2(n_2 + 1)}{2}, \tag{20}$$

Table 4: Comparison of HS, IHS, NGHS, and IGHS on six problems.

Problem	NI	Algorithm	ACT	Best	Worst	Mean	Std
Example 1a ($N = 5$)	2000	HS	1.659	1.1722$e-002$	3.5945$e-002$	1.9427$e-002$	8.0506$e-003$
		IHS	1.7011	8.4989$e-003$	2.6525$e-002$	1.6498$e-002$	5.9693$e-003$
		NGHS	1.781	5.9558$e-003$	3.0649$e-002$	1.2990$e-002$	7.8612$e-003$
		IGHS	2.6365	**5.4594e − 003**	**7.9210e − 003**	**5.9763e − 003**	**6.6092e − 004**
Example 1a ($N = 8$)	5000	HS	4.3975	2.9721$e-002$	9.3845$e-002$	5.4468$e-002$	1.5500$e-002$
		IHS	4.4121	3.1554$e-002$	8.1672$e-002$	5.1982$e-002$	1.4302$e-002$
		NGHS	4.4334	5.9619$e-003$	4.9489$e-002$	1.5350$e-002$	9.0064$e-003$
		IGHS	6.6589	**3.7074e − 003**	**1.0237e − 002**	**5.6536e − 003**	**1.7608e − 003**
Example 1b ($N = 5$)	2000	HS	0.57743	1.72096501$e-003$	4.06941185$e-003$	2.15386021$e-003$	5.99250994$e-004$
		IHS	0.56219	1.71809880$e-003$	1.82397928$e-003$	1.72847844$e-003$	2.71096253$e-005$
		NGHS	0.56886	1.71809919$e-003$	1.83200101$e-003$	1.72490455$e-003$	2.54479841$e-005$
		IGHS	0.84918	**1.71809876e − 003**	**1.71833108e − 003**	**1.71811926e − 003**	**5.18365165e − 008**
Example 1b ($N = 8$)	5000	HS	2.6544	1.7202$e-003$	2.3210$e-003$	1.8309$e-003$	1.5559$e-004$
		IHS	2.651	1.7119$e-003$	1.7721$e-003$	1.7194$e-003$	1.8286$e-005$
		NGHS	2.6787	1.7119$e-003$	1.8486$e-003$	1.7188$e-003$	3.0568$e-005$
		IGHS	4.0572	1.7119$e-003$	**1.7119e − 003**	**1.7119e − 003**	**1.4972e − 009**
Example 2a ($N = 8$)	5000	HS	5.3252	3.8006$e+000$	4.0102$e+000$	3.8879$e+000$	5.2382$e-002$
		IHS	5.2646	3.7901$e+000$	3.9328$e+000$	3.8548$e+000$	4.3294$e-002$
		NGHS	5.3327	3.6604$e+000$	3.7327$e+000$	3.6760$e+000$	1.6444$e-002$
		IGHS	7.9408	**3.6560e + 000**	**3.6628e + 000**	**3.6581e + 000**	**1.9936e − 003**
Example 2b ($N = 8$)	5000	HS	3.1802	1.1975$e+000$	1.4046$e+000$	1.2670$e+000$	5.0691$e-002$
		IHS	3.1454	1.1667$e+000$	1.3149$e+000$	1.2283$e+000$	3.8183$e-002$
		NGHS	3.1735	1.0260$e+000$	1.1026$e+000$	1.0445$e+000$	1.8699$e-002$
		IGHS	4.7435	**1.0200e + 000**	**1.0392e + 000**	**1.0277e + 000**	**5.3827e − 003**

where n_1 and n_2 denote the sizes of sample 1 and sample 2, respectively. R_1 and R_2 denote the sums of the ranks in sample 1 and sample 2, respectively, U_1 represents the number of sample 1 observations beaten by sample 2 observations, and U_2 represents the number of sample 2 observations beaten by sample 1 observations. By combining (19) and (20), the sum of U_1 and U_2 is calculated as follows:

$$U_1 + U_2 = R_1 - \frac{n_1(n_1+1)}{2} + R_2 - \frac{n_2(n_2+1)}{2}. \quad (21)$$

Based on the known conditions $R_1 + R_2 = n(n+1)/2$ and $n = n_1 + n_2$, we can easily obtain the simplified form of (21) as follows:

$$U_1 + U_2 = n_1 n_2. \quad (22)$$

To make the parameters R_1 and R_2 easy to understand, one simple and interesting example is provided as follows: a sample of 6 tortoises (sample 1) and 6 hares (sample 2) are collected and made run in a race. The order in which they reach the finishing post (their rank order, from first to last) is as follows: T H H H H H T T T T T H, where T represents a tortoise and H represents a hare.

A direct way: we take each tortoise in turn and count the number of hares it is beaten by (lower rank), getting 0, 5, 5, 5, 5, and 5, which means $U_1 = 0 + 5 + 5 + 5 + 5 + 5 = 25$. In the meantime, we could take each hare in turn and count the number of tortoises it is beaten by. In this situation, we get 1, 1, 1, 1, 1, and 6. So $U_2 = 6 + 1 + 1 + 1 + 1 + 1 = 11$. It is clear that the sum of these two values for U is 36, which is 6×6.

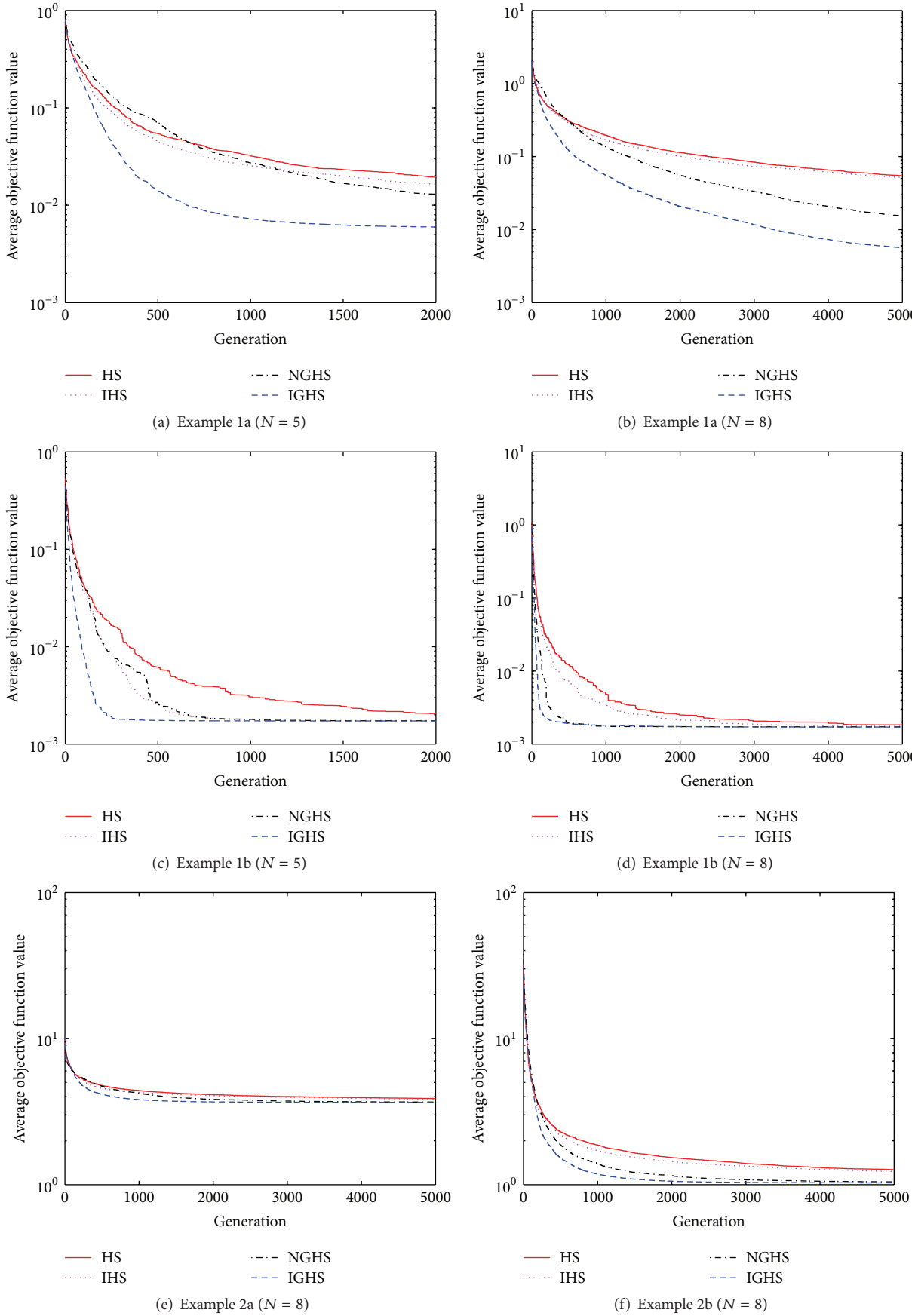
Table 5: Mann-Whitney U test result obtained using four methods.

Problem/algorithm	U_{HS}	U_{IHS}	U_{NGHS}
Example 1a ($N = 5$)	400	400	380
Example 1a ($N = 8$)	400	400	390
Example 1b ($N = 5$)	400	177	298
Example 1b ($N = 8$)	400	355	257
Example 2a ($N = 8$)	400	400	395
Example 2b ($N = 8$)	400	400	347

An indirect way: the sum of the ranks achieved by the tortoises is $R_1 = 1 + 7 + 8 + 9 + 10 + 11 = 46$, and thus $U_1 = 46 - (6 \times 7)/2 = 46 - 21 = 25$ (as in (19)). The sum of the ranks achieved by the hares is $R_2 = 2 + 3 + 4 + 5 + 6 + 12 = 32$, and thus $U_2 = 32 - 21 = 11$ (as in (20)).

In order to compare the IGHS with the other three HSs in a statistical way, three groups of Mann-Whitney U tests are executed, and they are (U_{HS}, U_{IGHS}), (U_{IHS}, U_{IGHS}), and (U_{NGHS}, U_{IGHS}), respectively. For each problem, 20 independent experiments are carried out; therefore, $n_{other} = n_{IGHS} = 20$ and $U_{other} + U_{IGHS} = n_{other} \times n_{IGHS} = 400$, where the subscript other denotes any of the other three HSs including the HS, IHS, and NGHS. The parameters of the four HSs are exactly the same as those of the aforementioned section, and Table 5 lists the results of Mann-Whitney U tests.

From Table 5, it is evident that the IGHS completely dominates the HS for solving all problems, because the values



(a) Example 1a ($N = 5$)

(b) Example 1a ($N = 8$)

(c) Example 1b ($N = 5$)

(d) Example 1b ($N = 8$)

(e) Example 2a ($N = 8$)

(f) Example 2b ($N = 8$)

Figure 8: Average convergence curves of four methods on solving six problems.

Table 6: The SNR values (in dB form) corresponding to various standard deviations.

Problem/algorithm	$\sigma = 0.001$	$\sigma = 0.002$	$\sigma = 0.005$	$\sigma = 0.01$	$\sigma = 0.02$	$\sigma = 0.05$	$\sigma = 0.1$	$\sigma = 0.2$
Example 1a ($N = 5$)	49.88	44.812	36.005	28.965	23.873	16.266	10.286	3.8217
Example 1a ($N = 8$)	49.567	43.463	35.865	29.496	24.641	15.688	10.307	3.911
Example 1b ($N = 5$)	51.415	46.475	37.818	31.1	25.635	17.753	12.19	5.296
Example 1b ($N = 8$)	51.448	44.949	37.822	31.875	25.634	19.216	11.73	5.2978
Example 2a ($N = 8$)	89.077	65.084	58.361	51.488	48.062	38.586	32.856	23.167
Example 2b ($N = 8$)	77.16	70.793	63.382	56.586	51.653	43.103	36.481	31.221

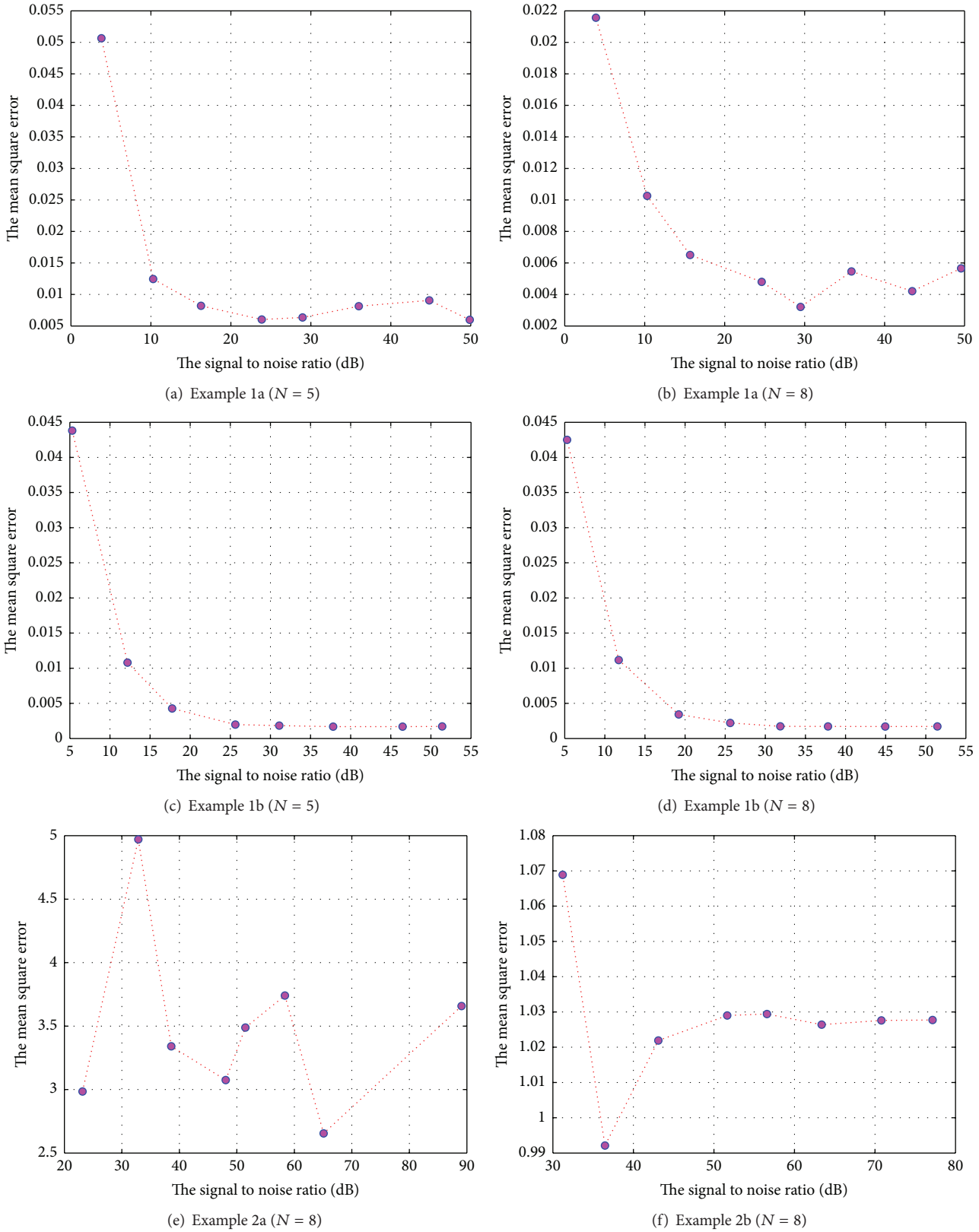
Table 7: Effects of various standard deviations on the average MSEs of HS, IHS, NGHS, and IGHS.

Problem	NI	Algorithm	$\sigma = 0.002$	$\sigma = 0.005$	$\sigma = 0.01$	$\sigma = 0.02$	$\sigma = 0.05$	$\sigma = 0.1$	$\sigma = 0.2$
Example 1a ($N = 5$)	2000	HS	0.031915	0.026339	0.021641	0.020161	0.031606	0.030344	0.073344
		IHS	0.023133	0.01932	0.018847	0.017164	0.025531	0.02863	0.06602
		NGHS	0.012373	0.01317	0.0080251	0.0087471	0.017701	0.015685	0.05541
		IGHS	**0.0090595**	**0.0081459**	**0.0063491**	**0.0060281**	**0.0082031**	**0.012456**	**0.05064**
Example 1a ($N = 8$)	5000	HS	0.04305	0.053554	0.044272	0.061806	0.052752	0.056594	0.072702
		IHS	0.040038	0.05833	0.039185	0.050786	0.059166	0.044153	0.065037
		NGHS	0.01394	0.017639	0.011319	0.016349	0.017154	0.022171	0.034087
		IGHS	**0.0042096**	**0.0054617**	**0.0032124**	**0.004798**	**0.0065079**	**0.01026**	**0.021561**
Example 1b ($N = 5$)	2000	HS	0.0020816	0.002023	0.0021274	0.0023366	0.0048039	0.011212	0.044087
		IHS	0.0016932	0.0016984	0.001846	0.0020514	0.0042728	0.010803	0.043885
		NGHS	0.0016883	0.0016907	0.0018335	0.0019744	0.004277	0.010804	0.043811
		IGHS	**0.0016881**	**0.0016894**	**0.0018321**	**0.0019734**	**0.0042728**	**0.010799**	**0.043809**
Example 1b ($N = 8$)	5000	HS	0.0021598	0.002027	0.0018595	0.0023973	0.0035314	0.011308	0.042578
		IHS	0.0017257	0.0017276	0.0017401	0.0022161	0.0034728	0.011198	0.042543
		NGHS	0.0017311	0.0017232	0.0017334	0.0022162	0.0034424	0.011172	0.0425
		IGHS	**0.0017063**	**0.0017185**	**0.0017295**	**0.0022099**	**0.0034249**	**0.011163**	**0.042491**
Example 2a ($N = 8$)	5000	HS	2.8627	3.9382	3.7033	3.2856	3.5386	5.1986	3.1728
		IHS	2.8343	3.9123	3.6921	3.2648	3.5163	5.1678	3.1443
		NGHS	2.6651	3.7442	3.5112	3.0952	3.3575	4.9886	2.9954
		IGHS	**2.6546**	**3.7404**	**3.4886**	**3.0747**	**3.3407**	**4.9715**	**2.9847**
Example 2b ($N = 8$)	5000	HS	1.2711	1.2759	1.2776	1.2868	1.292	1.2358	1.3175
		IHS	1.2339	1.236	1.2508	1.2473	1.242	1.2165	1.2836
		NGHS	1.0437	1.0418	1.0432	1.0411	1.0456	1.0064	1.0917
		IGHS	**1.0276**	**1.0264**	**1.0294**	**1.029**	**1.0219**	**0.99213**	**1.0689**

of U_{HS} are all equal to 400. In other words, the IGHS has never been beaten by the HS. Furthermore, the values of U_{NGHS} are significantly greater than 200, which indicates that the IGHS is also superior to the NGHS. In addition, the values of U_{IHS} are significantly greater than 200 for all problems except Example 1b ($N = 5$). Thus, the IGHS performs better than the IHS on solving five of the six problems. Regarding Example 1b ($N = 5$), the U_{IHS} value is equal to 177, which is smaller than but close to 200. Thus, the IHS is a little better than the IGHS for Example 1b ($N = 5$).

Figure 8 shows the average convergence curves of four methods over 20 runs for six problems. It is clear that the HS has the poorest performance, and its convergence rate is the slowest. In addition, both the IHS and the NGHS converge faster than the HS but slower than the IGHS. Obviously, the IGHS has the fastest convergence rate in each case. For Example 1a ($N = 5$) and ($N = 8$), it can achieve lower levels than the other three HSs. With respect to Example 2b ($N = 8$), the NGHS and the IGHS can converge to comparable levels, which are lower than those of the HS and the IHS. Moreover, the IHS, the NGHS, and the IGHS can reach comparable levels for the remaining cases. Strictly, the IGHS still outperforms the other three HSs for these cases according to Table 4.

In Example 1, the standard deviation of the Gaussian noise is so small ($\sigma = 0.001$) as it is equivalent to considering a negligible noise, while in Example 2 the simulations are carried out without noise. To investigate the robustness of the proposed algorithm with respect to the additive noise, different standard deviations (σ) of the Gaussian noise are considered when solving a problem of system identification. On the other hand, signal to noise ratio (SNR) is a measure used in science and engineering that compares the level of a desired signal to the level of background noise. In this paper, the amplitudes of signal and noise are measured, and thus the SNR value can be calculated by



(a) Example 1a ($N = 5$)

(b) Example 1a ($N = 8$)

(c) Example 1b ($N = 5$)

(d) Example 1b ($N = 8$)

(e) Example 2a ($N = 8$)

(f) Example 2b ($N = 8$)

FIGURE 9: The mean square error (MSE) versus the signal to noise ratio (SNR).

$\mathrm{SNR}_{\mathrm{dB}} = 20\log_{10}(A_{\mathrm{signal}}/A_{\mathrm{noise}})$, where A represents root mean square (RMS) amplitude. Given a standard deviation for a problem, a corresponding SNR value can be determined. Therefore, the SNR values related to various standard deviations are reported in Table 6.

From Table 6, it is clear that the SNR value decreases as the standard deviation of the Gaussian noise increases. Therefore, a Gaussian noise with larger σ value will exert more serious effects on signal than the one with smaller σ value, which is harmful to the extraction of signal. In addition, a plot of MSE versus SNR is given in Figure 9.

In Figures 9(a), 9(b), 9(c), and 9(d), the fluctuations of MSE are minor in the range $20 \leq \mathrm{SNR}_{\mathrm{dB}} \leq 55$. In Figure 9(f), the fluctuations of MSE are minor in the range $40 \leq \mathrm{SNR}_{\mathrm{dB}} \leq 80$. In Figure 9(e), the fluctuations of MSE are slightly larger than those of the above five examples. Moreover, the MSE values tend to get larger when SNR decreases in most cases. In this experiment, the standard deviation σ varies in a wide range $[0.001, 0.2]$, the MSE values remain low levels for most σ values, and the large fluctuations of MSE do not happen until σ increases to very high levels (such as $\sigma = 0.1, 0.2$). Therefore, the robustness of IGHS with respect to the additive noise is acceptable to some extent.

Various standard deviations of the Gaussian noise will have different effects on the convergence of HS, IHS, NGHS, and IGHS for six problems. To testify and compare the convergence of the proposed algorithm and the other three HSs, the average MSEs associated with the standard deviation are listed in Table 7.

The best results are marked in bold in Table 7. For any problem with various standard deviations, IGHS can always obtain the smallest MSEs among four algorithms, indicating that IGHS has stronger convergence and stability than the other three HSs. Overall, IGHS has exhibited more desirable robustness to the Gaussian noise than the other three HSs for all six problems.

6. Conclusions

In the present paper, we propose an improved NGHS algorithm, called IGHS, for identifying nonlinear discrete-time systems based on second-order Volterra model. The prime objective of the IGHS is to attain the most appropriate kernels of Volterra model so that the estimated output can track and characterize the actual output as much as possible. Furthermore, we use two examples with different input signals and system memory sizes to test the performance of the IGHS. Experimental results suggest that the IGHS performs well and is superior to the other three methods in most cases according to four criteria ("Best," "Worst," "Mean," and "Std") associated with the mean square error (MSE). Thus, the IGHS is a strong candidate for the identification of nonlinear system based on second-order Volterra model.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work is supported by Jiangsu Province Science and Technology Support program (industry), no. BE2014045.

References

[1] R. J. G. B. Campello, W. C. do Amaral, and G. Favier, "A note on the optimal expansion of Volterra models using Laguerre functions," *Automatica*, vol. 42, no. 4, pp. 689–693, 2006.

[2] M. Masugi and T. Takuma, "Using a Volterra system model to analyze nonlinear response in video-packet transmission over IP networks," *Communications in Nonlinear Science and Numerical Simulation*, vol. 12, no. 3, pp. 411–421, 2007.

[3] J. K. Gruber, J. L. Guzmán, F. Rodríguez, C. Bordons, M. Berenguel, and J. A. Sánchez, "Nonlinear MPC based on a Volterra series model for greenhouse temperature control using natural ventilation," *Control Engineering Practice*, vol. 19, no. 4, pp. 354–366, 2011.

[4] I. Tawfiq and T. Vinh, "Nonlinear behaviour of structures using the Volterra series—signal processing and testing methods," *Nonlinear Dynamics*, vol. 37, no. 2, pp. 129–149, 2004.

[5] K. Worden and G. Manson, "A Volterra series approximation to the coherence of the Duffing oscillator," *Journal of Sound and Vibration*, vol. 286, no. 3, pp. 529–547, 2005.

[6] L. M. Li and S. A. Billings, "Piecewise Volterra modeling of the Duffing oscillator in the frequency-domain," *Mechanical Systems and Signal Processing*, vol. 26, no. 1, pp. 117–127, 2012.

[7] W.-D. Chang, "Volterra filter modeling of nonlinear discrete-time system using improved particle swarm optimization," *Digital Signal Processing*, vol. 22, no. 6, pp. 1056–1062, 2012.

[8] C.-H. Cheng and E. J. Powers, "Optimal Volterra kernel estimation algorithms for a nonlinear communication system for PSK and QAM inputs," *IEEE Transactions on Signal Processing*, vol. 49, no. 1, pp. 147–163, 2001.

[9] E. E. Kuruoğlu, "Nonlinear least l_p-norm filters for nonlinear autoregressive α-stable processes," *Digital Signal Processing*, vol. 12, no. 1, pp. 119–142, 2002.

[10] S.-W. Nam and E. J. Powers, "Volterra series representation of time-frequency distributions," *IEEE Transactions on Signal Processing*, vol. 51, no. 6, pp. 1532–1537, 2003.

[11] G. Liniin and S. Puthusserypady, "Performance analysis of volterra-based nonlinear adaptive blind multiuser detectors for DS-CDMA systems," *Signal Processing*, vol. 84, no. 10, pp. 1941–1956, 2004.

[12] A. Y. Kibangou, G. Favier, and M. M. Hassani, "Selection of generalized orthonormal bases for second-order Volterra filters," *Signal Processing*, vol. 85, no. 12, pp. 2371–2385, 2005.

[13] C. Krall, K. Witrisal, G. Leus, and H. Koeppl, "Minimum mean-square error equalization for second-order Volterra systems," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4729–4737, 2008.

[14] H. Tang, Y. H. Liao, J. Y. Cao, and H. Xie, "Fault diagnosis approach based on Volterra models," *Mechanical Systems and Signal Processing*, vol. 24, no. 4, pp. 1099–1113, 2010.

[15] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[16] D. X. Zou, L. Q. Gao, J. H. Wu, S. Li, and Y. Li, "A novel global harmony search algorithm for reliability problems," *Computers & Industrial Engineering*, vol. 58, no. 2, pp. 307–316, 2010.

[17] N. Poursalehi, A. Zolfaghari, A. Minuchehr, and K. Valavi, "Self-adaptive global best harmony search algorithm applied to reactor core fuel management optimization," *Annals of Nuclear Energy*, vol. 62, pp. 86–102, 2013.

[18] S. O. Degertekin, "Improved harmony search algorithms for sizing optimization of truss structures," *Computers & Structures*, vol. 92-93, pp. 229–241, 2012.

[19] J. S. Zhang and H. Q. Zhao, "A novel adaptive bilinear filter based on pipelined architecture," *Digital Signal Processing*, vol. 20, no. 1, pp. 23–38, 2010.

[20] D. X. Zou, L. Q. Gao, J. H. Wu, and S. Li, "Novel global harmony search algorithm for unconstrained problems," *Neurocomputing*, vol. 73, no. 16–18, pp. 3308–3318, 2010.

[21] J. Kennedy, "Bare bones particle swarms," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 80–87, Indianapolis, Ind, USA, April 2003.

[22] J. Q. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[23] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or Cauchy jumps," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 3285–3291, IEEE, Trondheim, Norway, May 2009.

[24] H. R. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," in *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 1, pp. 695–701, IEEE, Vienna, Austria, November 2005.

[25] R. R. Sumar, A. A. R. Coelho, and L. Dos Santos Coelho, "Computational intelligence approach to PID controller design using the universal model," *Information Sciences*, vol. 180, no. 20, pp. 3980–3991, 2010.

[26] M. Mahdavi, M. Fesanghary, and E. Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied Mathematics and Computation*, vol. 188, no. 2, pp. 1567–1579, 2007.

[27] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[28] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Annals of Mathematical Statistics*, vol. 18, pp. 50–60, 1947.