Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Volterra filter modeling of a nonlinear discrete-time system based on a ranked differential evolution algorithm*

De-xuan ZOU†[1], Li-qun GAO[2], Steven LI[3]

([1]*School of Electrical Engineering and Automation, Jiangsu Normal University, Xuzhou 221116, China*)

([2]*School of Information Science and Engineering, Northeastern University, Shenyang 110004, China*)

([3]*Graduate School of Business and Law, RMIT University, GPO Box 2476, Melbourne 3001, Australia*)

†E-mail: zoudexuan@163.com

Received Dec. 6, 2013; Revision accepted Jan. 27, 2014; Crosschecked July 16, 2014

Abstract: This paper presents a ranked differential evolution (RDE) algorithm for solving the identification problem of nonlinear discrete-time systems based on a Volterra filter model. In the improved method, a scale factor, generated by combining a sine function and randomness, effectively keeps a balance between the global search and the local search. Also, the mutation operation is modified after ranking all candidate solutions of the population to help avoid the occurrence of premature convergence. Finally, two examples including a highly nonlinear discrete-time rational system and a real heat exchanger are used to evaluate the performance of the RDE algorithm and five other approaches. Numerical experiments and comparisons demonstrate that the RDE algorithm performs better than the other approaches in most cases.

Key words: Ranked differential evolution, Identification problem, Nonlinear discrete-time systems, Volterra filter model, Premature convergence

doi:10.1631/jzus.C1300350 **Document code:** A **CLC number:** TN713$^+$.7

1 Introduction

The Volterra filter model (Chang, 2012) belongs to the category of polynomial filters, and can approximate many real nonlinear systems. One significant advantage of the Volterra model is that its output is a linear combination of nonlinear functions of the input signal and linearly depends on the model coefficient known as the kernel. The Volterra filter model can be found in a wide variety of application areas including the identification of a parametric loudspeaker system (Ji and Gan, 2012), nonlinear acoustic echo cancellation (Contan *et al*., 2013), the modeling of the oscillation behavior of ultrasound contrast agents (Mleczko *et al*., 2009), and other areas (Kuruoğlu, 2002; Nam and Powers, 2003; Krall *et al*., 2008; Tang *et al*., 2010). Ji and Gan (2012) developed a nonlinear system identification model based on an adaptive Volterra filter in order to realize the identification of a parametric loudspeaker system. Unlike a conventional loudspeaker, the nonlinear characteristic of a parametric loudspeaker system is determined by some parameters in nonlinear acoustics, namely the initial pressure of the primary waves, observing distance and angle, ambient temperature, and relative humidity. By using a truncated Volterra series up to the second-order kernel, the sound pressure level and harmonic distortion can be accurately predicted. Contan *et al*. (2013) combined a modified version of the normalized least-mean-fourth (NLMF) algorithm and an adaptive second-order Volterra structure for acoustic echo cancellation, which reaches a compromise between convergence rate and steady-state error. Furthermore, a convergence rate improvement was achieved at the same steady-state error by modifying the step size of the conventional NLMF algorithm

* Project supported by the Science Fundamental Research Project of Jiangsu Normal University, China (No. 9212812101)

based on a new step-size function. Mleczko *et al*. (2009) discussed and evaluated the use of Volterra series for modeling the scattering behavior of contrast agent microbubbles. For moderate insonification pressures, a Volterra series enables an adequate representation of the oscillation behavior of ultrasound contrast agents. The accuracy of the evaluated model was satisfactory for insonification pressures up to 100 kPa. Cheng and Powers (2001) presented a fifth-order Volterra kernel estimation algorithm for a bandpass nonlinear system under uniformly independent and identically distributed (i.i.d.) rectangular M-ary quadrature amplitude modulation (M-QAM) input and uniformly i.i.d. M-ary phase shift keying (M-PSK) input. The fifth-order Volterra kernel algorithm is able to readily capture the behavior of a nonlinear system.

Besides the Volterra filter model we studied the evolutionary algorithm. Chang (2012) proposed an improved particle swarm optimization (IPSO) to determine the kernel vector of the Volterra filter model. In this paper, we propose a ranked differential evolution (RDE) algorithm, and test it as an alternative to the IPSO. The differential evolution (DE) algorithm, derived by Storn and Price (1995), has attracted the interest of many researchers, because it has a simple algorithm structure and a favorable optimizing performance. In recent years, many improved versions of the DE algorithm have been used in a variety of engineering fields, including the task assignment problem (Zou *et al*., 2011), optimization of nonlinear chemical processes (Babu and Angira, 2006), and optimal synthesis of linear antenna arrays (Li and Yin, 2012).

2 Volterra filter model and its truncated second-order form

The Volterra filter model plays an important role in identifying unknown nonlinear systems, and has drawn much attention from researchers in recent decades. Generally, the discrete form of the Volterra filter model of the qth order (Chang, 2012) is stated as

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

where N stands for the system memory size. Eq. (1) denotes the Volterra filter model with the infinite series. In this paper, we study only the truncated second-order Volterra model (Zhang and Zhao, 2010; Chang, 2012) as follows:

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

For simplicity, Eq. (2) can be expressed as the following vector form:

$$y[n]=\boldsymbol{H}\boldsymbol{X}^{\mathrm{T}}. \tag{3}$$

Here $\boldsymbol{H}$ stands for the Volterra kernel vector given by

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

and $\boldsymbol{X}$ denotes the Volterra input vector given by

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

According to Eqs. (4) and (5), $\boldsymbol{H}$ and $\boldsymbol{X}$ have the same vector length which can be calculated as follows (Zhang and Zhao, 2010):

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

To enable the output $y[n]$ to approximate the actual system output as much as possible, an RDE algorithm is proposed to optimize the variables of the kernel vector $\boldsymbol{H}$ under the input vector $\boldsymbol{X}$. More details of the RDE algorithm will be illustrated in Section 3. To understand better the working principle of RDE, it is necessary first to introduce the original DE algorithm.

3 Original differential evolution algorithm

DE (Storn and Price, 1995) is a simple evolutionary algorithm which produces new candidate solutions by combining parent individuals and some other individuals. A candidate solution replaces its parent solution only if it has a better objective function value or fitness. Generally speaking, DE works as follows:

Step 1: initialization of parameters and population

Initialize scale factor F, crossover rate CR, population size PS, the number of iterations NI, and the number of problem variables L. For the jth problem variable (j=1, 2, …, L), its lower and upper bounds are $\underline{x}_j$ and $\overline{x}_j$, respectively. Moreover, all the candidate solutions in the population are generated randomly from a uniform distribution in the range $[\underline{x}_j, \overline{x}_j]$.

Step 2: mutation

A trial vector $\boldsymbol{v}_i^{t+1}$ is produced by mutating a target vector. Usually, the updating equation of the trial vector $\boldsymbol{v}_i^{t+1}$ is given by

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{x}_{i_3}^t + F(\boldsymbol{x}_{i_1}^t - \boldsymbol{x}_{i_2}^t). \tag{7}$$

Here, t represents the index of the current iteration. F ($F\in[0, 2]$) is a scale factor which affects the differential variation between two candidate solutions. In addition, i_1, i_2, and i_3 are three different integers which are randomly chosen from the set {1, 2, …, PS}.

Step 3: crossover

The variables of offspring vector $\boldsymbol{u}_i^{t+1}$ are the combination of parent vector $\boldsymbol{x}_i^t$ and trial vector $\boldsymbol{v}_i^{t+1}$. Thus, they are calculated as follows:

$$u_{i,j}^{t+1} = \begin{cases} v_{i,j}^{t+1}, & \text{if rand}() \le \text{CR or } j = j_r, \\ x_{i,j}^t, & \text{otherwise,} \end{cases} \tag{8}$$

where rand() is a randomly generated number from [0, 1], j_r is a randomly chosen index from [1, L], and CR (CR$\in$[0, 1]) stands for the crossover rate.

Step 4: selection

If the fitness of $\boldsymbol{u}_i^{t+1}$ is better than that of $\boldsymbol{x}_i^t$, offspring vector $\boldsymbol{u}_i^{t+1}$ is adopted for $\boldsymbol{x}_i^{t+1}$; otherwise, the parent vector $\boldsymbol{x}_i^t$ is chosen for $\boldsymbol{x}_i^{t+1}$. Therefore, the selection operation is given by

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^{t+1}, & \text{if } f(\boldsymbol{u}_i^{t+1}) < f(\boldsymbol{x}_i^t), \\ \boldsymbol{x}_i^t, & \text{otherwise.} \end{cases} \tag{9}$$

Step 5: check the stopping criterion

If the number of iterations (NI) is reached, the DE procedure is stopped. Otherwise, steps 3 and 4 are repeated.

4 A ranked differential evolution algorithm

To perform the system identification accurately, in this section we propose an RDE algorithm for optimally determining the Volterra kernel vector. More specifically, the RDE algorithm makes two improvements on DE algorithms as follows:

1. The modification of the scale factor F. The RDE algorithm adjusts the scale factor F by combining a sine function and randomness, and the new scale factor can be expressed as follows:

$$F^t = [1+\sin(\omega_0 t)]\times\text{rand}(). \tag{10}$$

Here, t=1, 2, …, NI is the index of the current iteration, and rand() denotes a random number in [0, 1], which is helpful to increase the diversity of the population. $\omega_0 = 2\pi/T_0$ is the radian frequency of the sine function, and T_0 is its corresponding period. Suppose NI=150 and T_0=NI/10, then scale factor F can be described as in Fig. 1.

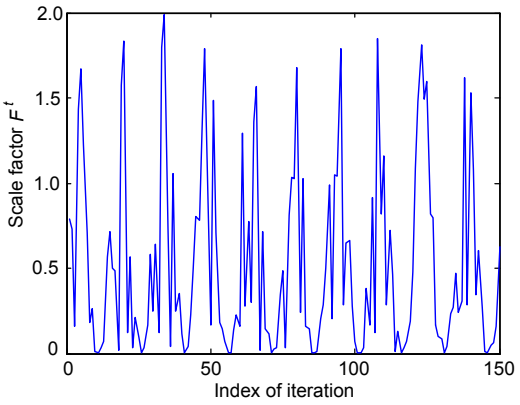

Fig. 1 Scale factor during 150 iterations

The scale factor F^t is sometimes small, which is beneficial for performing a local search, and sometimes large, which is beneficial for performing a global search. Moreover, both small and large values exist in each fixed iteration interval (T_0), which ensures that the RDE algorithm has both global and local searching capacity in the whole evolution process.

2. The RDE algorithm ranks all the candidate solutions of the population according to their objective function values. The smaller (better) the objective function value of a candidate solution, the higher its rank. After ranking all the candidate solutions, a modified mutation operation is performed, given by

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{x}_{i_c}^t + F^t(\boldsymbol{x}_{i_a}^t - \boldsymbol{x}_{i_b}^t). \tag{11}$$

Here, i_c represents the index randomly chosen from the set $\{1, 2, \ldots, \mathrm{PS}\}$, i_a represents the index randomly chosen from the set $\{1, 2, \ldots, \mathrm{PS}/2\}$, i_b represents the index randomly chosen from the set $\{\mathrm{PS}/2+1, \ldots, \mathrm{PS}\}$, and $i_a \neq i_b \neq i_c \neq i$. Obviously, $\boldsymbol{x}_{i_a}^t$ comes from the candidate solutions with better objective function values, and $\boldsymbol{x}_{i_b}^t$ comes from those with worse values. By using this selection mechanism, the searching range of difference vector $(\boldsymbol{x}_{i_a}^t - \boldsymbol{x}_{i_b}^t)$ can be further enlarged, which can effectively avoid premature convergence of the RDE algorithm.

Based on the above two modifications, the performance of the RDE algorithm can be improved. Its procedure is illustrated in Algorithm 1.

Adopting the RDE algorithm, a schematic diagram of nonlinear discrete-time systems based on the truncated second-order Volterra model is shown in Fig. 2, and the notations used in the diagram are listed in Table 1.

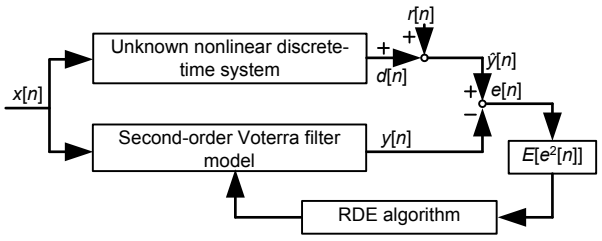

Fig. 2 Volterra filter modeling of a nonlinear discrete-time system using the RDE algorithm

The Volterra filter model based on the RDE algorithm aims to determine the optimal kernel vector $\boldsymbol{H}$ so that the output $y[n]$ approximates the actual

Algorithm 1 Ranked differential evolution

Begin
Set PS=50; NI=150; T_0=NI/10; $\underline{\boldsymbol{x}} = (\underline{x}_1, \underline{x}_2, \cdots, \underline{x}_L)$; $\overline{\boldsymbol{x}} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_L)$; CR=0.3
Randomly generate a population
For t=1 to NI
  Calculate scale factor F^t according to Eq. (10)
  Rank the population according to the objective function values of all candidate solutions
  For i=1 to PS
    Randomly generate an integer i_a in the range [1, PS/2]
    Randomly generate an integer i_b in the range [PS/2+1, PS]
    Randomly generate an integer i_c in the range [1, PS], and $i_a \neq i_b \neq i_c \neq i$
    $\boldsymbol{v}_i^{t+1} = \boldsymbol{x}_{i_c}^t + F^t(\boldsymbol{x}_{i_a}^t - \boldsymbol{x}_{i_b}^t)$
    Randomly generate an integer j_r in the range [1, L]
    For j=1 to L
      If rand()<CR or $j=j_r$
        $u_{i,j}^{t+1} = v_{i,j}^{t+1}$
      Else
        $u_{i,j}^{t+1} = x_{i,j}^t$
      End If
    End For
    If $f(\boldsymbol{u}_i^{t+1}) < f(\boldsymbol{x}_i^t)$
      $\boldsymbol{x}_i^{t+1} = \boldsymbol{u}_i^{t+1}$
    Else
      $\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t$
    End If
  End For
End For
End

Table 1 Notations used in Volterra filter modeling of a nonlinear discrete-time system

Notation	Meaning
$x[n]$	Digital input signal
$y[n]$	Output signal of the second-order Volterra filter model (Eq. (2))
$d[n]$	Output of the unknown nonlinear discrete-time system
$r[n]$	Measurement noise
$\hat{y}[n]$	Sum of $d[n]$ and $r[n]$
$e[n]$	Error signal between $\hat{y}[n]$ and $y[n]$

system output $\hat{y}[n]$ as much as possible. In other words, the difference between $y[n]$ and $\hat{y}[n]$ should be minimized. Therefore, it is necessary to find an objective function to achieve the modeling requirement. In this paper, the objective function (Chang, 2012) is

$$E[e^2[n]]=\frac{1}{T}\sum_{n-0}^{T-1}e^2[n]=\frac{1}{T}\sum_{n-0}^{T-1}\left[\hat{y}[n]-y[n]\right]^2. \quad (12)$$

Here, $E[e^2[n]]$ is defined as the mean square error (MSE), and T denotes the sampling number. By utilizing the RDE algorithm to minimize the MSE, we can find the optimal kernel vector of the second-order Volterra model.

5 Experimental results and analysis

To test the applicability of the RDE algorithm for nonlinear system identification based on the Volterra filter model, two examples were chosen: a highly nonlinear discrete-time rational system and a real heat exchanger.

5.1 Example 1

The first example was a highly nonlinear discrete-time rational system (Chang, 2012). Its mathematical model is stated as follows:

$$d[n]=\frac{0.3d^2[n-1]+0.8x[n-1]+0.6d[n-2]}{1+x^2[n-1]+d^2[n-1]}. \quad (13)$$

Two kinds of input signals were used. The first (Example 1a) was a random sequence uniformly generated from [−1, 1], which can be expressed as $x[n]\sim U(-1, 1)$, and the second (Example 1b) was a cosine signal $x[n]=0.8\cos(n\pi/9)$. In addition, the measurement noise $r[n]$ was assumed to be a Gaussian noise of $N(0, 0.001)$.

In this experiment, the RDE algorithm was used for solving Example 1a with $x[n]\sim U(-1, 1)$ and N=5. The parameters of the algorithm were set as follows: CR=0.3, PS=50, NI=150, the period of sine function T_0=NI/10, and sampling number T=100. Based on the above parameter settings, the estimated output $y[n]$ was obtained (Fig. 3).

In Fig. 3, a comparison between the actual output $\hat{y}[n]$ and the Volterra model output $y[n]$ is plotted with respect to the sampling number n. It is clear that the estimated output $y[n]$ was very close to the actual output $\hat{y}[n]$. Moreover, the final objective function value, i.e., mean square error (MSE), obtained using the RDE algorithm was equal to −44.136 dB. To verify the effectiveness of the RDE algorithm for Volterra filter modeling with a large memory size N, we increased N to 8. In Example 1a with $x[n]\sim U(-1, 1)$ and N=8, we also used the RDE algorithm to realize nonlinear system identification, and the parameters of the algorithm were set as follows: CR=0.3, PS=80, NI=200, the period of sine function T_0=NI/10, and sampling number T=100. Based on these parameter settings, the estimated output $y[n]$ was as shown in Fig. 4.

The estimated output $y[n]$ was very close to the actual output $\hat{y}[n]$ (Fig. 4). The MSE obtained using the RDE algorithm was −48.9 dB.

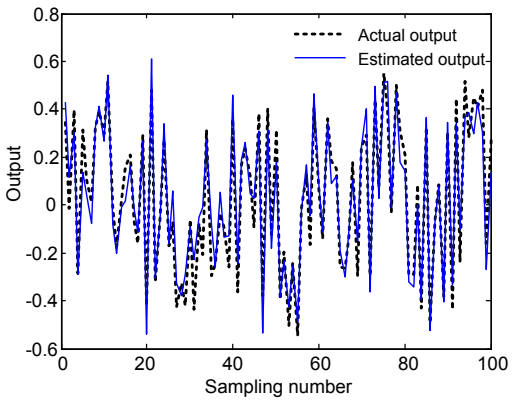

Fig. 3 Comparison of actual system output and Volterra model output for Example 1a (N=5)

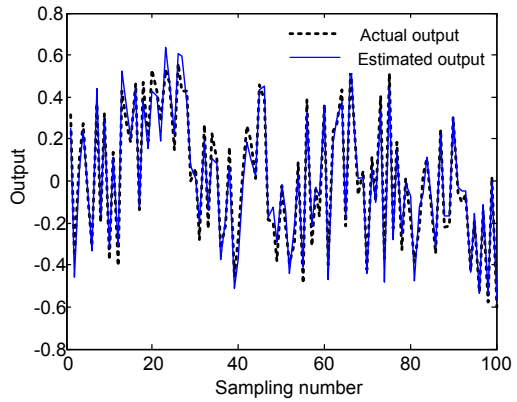

Fig. 4 Comparison of actual system output and Volterra model output for Example 1a (N=8)

To testify further the performance of Volterra filter modeling based on RDE, another testing input signal $x[n]$=0.8cos($n\pi$/9) (Example 1b) was adopted. In Example 1b with N=5 and N=8, the parameter settings of the RDE algorithm were the same as those for Example 1a. Figs. 5 and 6 depict the experimental results when N=5 and N=8, respectively. Two satisfying MSEs, –54.789 dB for N=5 and –54.849 for N=8, were obtained after performing the RDE algorithm.

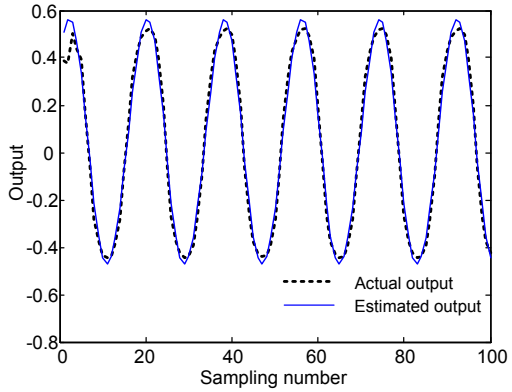

Fig. 5 Comparison of actual system output and Volterra model output for Example 1b (N=5)

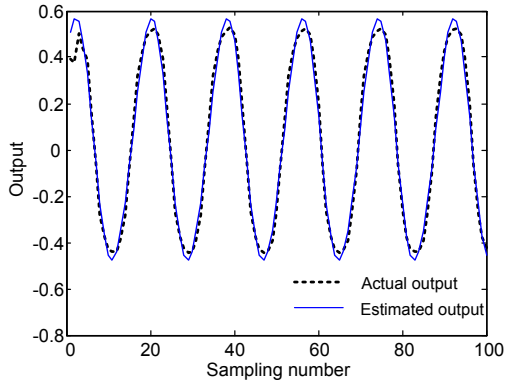

Fig. 6 Comparison of actual system output and Volterra model output for Example 1b (N=8)

5.2 Example 2

The second example was a real heat exchanger. Its mathematical model (Sumar *et al.*, 2010; Chang, 2012) is given by

$$w[n]=x[n]-1.3228x^2[n]+0.7671x^3[n]-2.1755x^4[n], \quad (14)$$

$$d[n]=\frac{-6.5306z^{-1}+5.5652z^{-2}}{1-1.608z^{-1}+0.6385z^{-2}}w[n], \quad (15)$$

where $x[n]$ is the process input denoting the flow rate and is constrained by the range [0, 1], $w[n]$ represents the static nonlinearity, and $d[n]$ represents the process output temperature. For convenience, Eqs. (14) and (15) can be further rewritten as the following difference equation:

$$d[n]=1.608d[n-1]-0.6385d[n-2]-6.5306w[n-1]+5.5652w[n-2]. \quad (16)$$

Two kinds of input signals were used. The first (Example 2a) was randomly generated in the range [0.1, 0.9], which can be expressed as $x[n]\sim U(0.1, 0.9)$, and the second (Example 2b) was a sine signal $x[n]$= 0.4sin($n\pi$/6)+0.5. The measurement noise $r[n]$=0.

In this experiment, the memory size of the Volterra filter model was set as N=8. To satisfy the physical input requirement, the system input $x[n]$ was randomly generated in the range [0.1, 0.9] and the testing input was then set to $x[n]$=0.4sin($n\pi$/6)+0.5. The parameters of the RDE algorithm for Example 2 were the same as those of Example 1. From performing the RDE algorithm, the Volterra model output for the modeling input is shown in Fig. 7 and for the testing input is shown in Fig. 8. Obviously, the experimental results are acceptable, and the Volterra model output $y[n]$ was close to the actual output $\hat{y}[n]$ in each case.

5.3 Comparison of five approaches for identifying nonlinear discrete-time systems

The RDE algorithm in this paper was compared with five other approaches for solving the above problems with different input signals and system memory sizes. The five other approaches were: particle swarm optimization (PSO) algorithm (Kennedy and Eberhart, 1995; Shi and Eberhart, 1999), improved particle swarm optimization (IPSO) (Chang, 2012), differential evolution (DE) (Storn and Price, 1995), improved differential evolution (IDE) algorithm (Zou *et al.*, 2011), and the differential evolution algorithm based on self-adapting control parameters (SADE) (Brest *et al.*, 2006). The parameters of all six approaches are listed in Table 2. The population size PS was set as 50 for the system memory size N=5 and as 80 for N=8. The number of iterations NI was set as 150 for the system memory size N=5 and as 200 for N=8. We conducted additional experiments with some

combinations with PS and NI using larger values: (PS=100, NI=250), (PS=120, NI=300), (PS=150, NI=350), and (PS=200, NI=400), and did not notice any significant differences in the results. We also conducted additional experiments with some combinations with PS and NI using smaller values: (PS=20, NI=50), (PS=30, NI=60), and (PS=40, NI=80). Although these parameter settings can save calculation time, they cannot guarantee calculation accuracy. Therefore, as the peak occurred at (PS=50, NI=150) for N=5 and at (PS=80, NI=200) for N=8, those values were used in this study. In addition, the sampling number was set as T=100 for the Volterra filter model. Matlab 7.0 was used to perform the above design steps in the environment of Intel Core i5-2410M CPU@2.30 GHz. Twenty independent runs were carried out for each problem, and the optimization results over 20 runs on each problem are reported in Table 3.

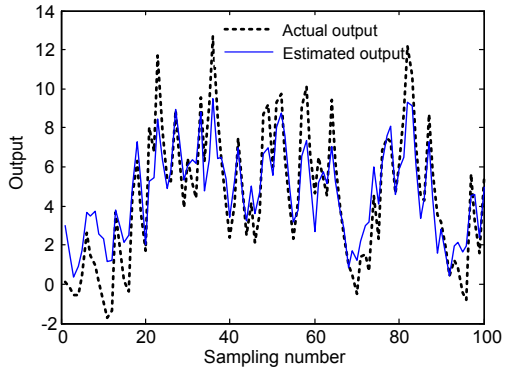

Fig. 7 Comparison of actual system output and Volterra model output for Example 2a (N=8)

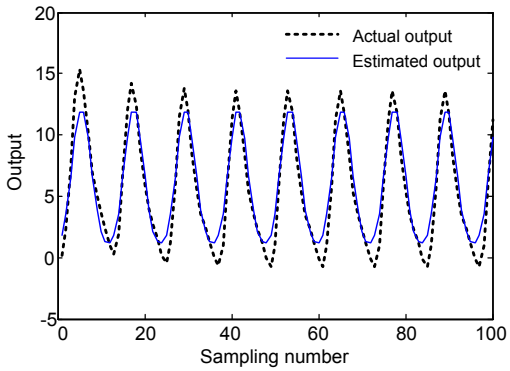

Fig. 8 Comparison of actual system output and Volterra model output for Example 2b (N=8)

The RDE algorithm showed overwhelming superiority over the other five approaches. The optimization results (Best, Worst, Mean, and Std) obtained using the RDE algorithm were better than those obtained by the other five approaches for Example 1a (N=5, 8), Example 2a (N=8), and Example 2b (N=8). In Example 1b (N=5), the 'Best' and 'Mean' results of the RDE algorithm were better than those of the other five approaches, and the "Worst" and 'Std' results of the DE algorithms were better than those of the other five approaches. In Example 1b (N=8), IPSO achieved the optimal solution, but was worse than the RDE algorithm according to the three criteria 'Worst', 'Mean', and 'Std'. The average computation times (ACTs) of the PSO, DE, IDE, SADE, and RDE algorithms were similar, but the ACT of IPSO was almost twice as long as any of the other five algorithms.

To determine whether the results produced by the RDE algorithm were statistically different from those produced by the other five approaches, Wilcoxon rank-sum tests (Wilcoxon, 1945; Derrac *et al.*, 2011) were conducted at the 5% significance level. The results are shown in Table 4. A P-value smaller than 0.05 suggests that the performance of the two approaches was statistically different with 95% certainty, whereas a P-value larger than 0.05 indicates no statistical difference.

In this experiment, the term 'Mean' was the primary criterion to define the best approach, and the term 'Std' ranked second. The results show that the RDE algorithm performed better than the other five approaches in solving all six problems. For Example 1b

Table 2 The parameter settings of six approaches

Approach	Parameters
Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995; Shi and Eberhart, 1999)	ω_{min}=0.4, ω_{max}=0.9, c_1=c_2=2
Improved particle swarm optimization (IPSO) (Chang, 2012)	ω=0.8, c_1=c_2=c_3=1, S=5
Differential evolution (DE) (Storn and Price, 1995)	F=0.6, CR=0.3
Improved differential evolution (IDE) (Zou *et al.*, 2011)	CR_{min}=0.1, CR_{max}=0.9
Differential evolution algorithm based on self-adapting control parameters (SADE) (Brest *et al.*, 2006)	F_l=0.1, F_u=0.9, τ_1=τ_2=0.1
Ranked differential evolution (RDE)	CR=0.3, T_0=NI/10

Table 3 Comparison of results for PSO, IPSO, DE, IDE, SADE, and RDE algorithms applied to six problems

Problem	PS	NI	Algorithm	ACT (s)	Best (dB)	Worst (dB)	Mean (dB)	Std (dB)
Example 1a (N=5)	50	150	PSO	1.7916	–42.6590	–15.5670	–25.9630	–26.3320
			IPSO	3.4897	–43.2500	–34.1800	–38.0310	–44.5380
			DE	1.7125	–44.1290	–44.1210	–44.1250	–115.9600
			IDE	1.7157	–44.1340	–44.1270	–44.1300	–116.7600
			SADE	1.7423	–44.1340	–43.7190	–44.0980	–83.5150
			RDE	1.7281	**–44.1360**	**–44.1350**	**–44.1350**	**–139.0500**
Example 1a (N=8)	80	200	PSO	6.2010	–40.0160	–6.3917	–14.7800	–19.0170
			IPSO	12.3145	–45.2590	–32.1300	–35.2460	–42.1260
			DE	6.0910	–47.8220	–46.5430	–47.3670	–75.5250
			IDE	6.0593	–48.2320	–47.5850	–47.8600	–81.6940
			SADE	6.0846	–48.7810	–45.9320	–47.7500	–68.1050
			RDE	6.1001	**–48.9000**	**–48.6240**	**–48.8070**	**–89.7550**
Example 1b (N=5)	50	150	PSO	2.9186	–54.3010	–16.3100	–31.9040	–26.3050
			IPSO	5.7728	–54.3010	–37.3390	–50.1510	–52.0300
			DE	2.8454	–54.6740	**–53.9080**	–54.2790	**–87.7270**
			IDE	2.8339	–54.6820	–53.7790	–54.3530	–85.9460
			SADE	2.8475	–54.6620	–53.2450	–54.3210	–82.2410
			RDE	2.8490	**–54.7890**	–53.5320	**–54.5110**	–83.0720
Example 1b (N=8)	80	200	PSO	10.9023	–54.3370	–32.5260	–46.4280	–44.9610
			IPSO	21.7446	**–54.8860**	–49.8960	–52.8670	–69.3360
			DE	10.8377	–54.7300	–54.4610	–54.6010	–95.3990
			IDE	10.7828	–54.8270	–54.5950	–54.7230	–96.2800
			SADE	10.7593	–54.8400	–54.4800	–54.7110	–93.7530
			RDE	10.8645	–54.8490	**–54.6640**	**–54.7560**	**–100.2900**
Example 2a (N=8)	80	200	PSO	7.6237	13.2810	14.2760	13.7040	–15.9570
			IPSO	15.5188	12.8670	14.6830	13.5110	–12.2280
			DE	7.8295	12.8090	13.0220	12.9140	–30.4490
			IDE	7.9206	12.7390	13.0260	12.8690	–28.9630
			SADE	7.7898	12.6230	12.8880	12.7350	–27.9690
			RDE	7.9615	**12.6100**	**12.7160**	**12.6500**	**–38.8710**
Example 2b (N=8)	80	200	PSO	12.0401	1.3429	10.5120	5.7152	–5.1475
			IPSO	23.9678	1.4398	5.7457	3.3947	–13.3590
			DE	11.9376	1.2283	1.9472	1.6248	–30.6170
			IDE	11.9065	0.9999	1.9701	1.5263	–28.8860
			SADE	11.9308	0.5966	1.6658	1.0861	–28.3250
			RDE	11.9023	**0.4155**	**0.8089**	**0.6556**	**–37.4780**

Act: average computation time; Std: standard deviation of 20 runs. Bold font signifies the best result among the six approaches

Table 4 *P*-values from Wilcoxon rank-sum tests of performance results for six problems

Problem	*P*-value					
	PSO	IPSO	DE	IDE	SADE	RDE
Example 1a (N=5)	6.7956×10^{-8}	6.6909×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	NA
Example 1a (N=8)	6.7956×10^{-8}	6.7765×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	3.4156×10^{-7}	NA
Example 1b (N=5)	1.9177×10^{-7}	1.6571×10^{-7}	2.7451×10^{-4}	3.0566×10^{-3}	7.1135×10^{-3}	NA
Example 1b (N=8)	6.7956×10^{-8}	1.2009×10^{-6}	5.2269×10^{-7}	1.9883×10^{-1}	2.8530×10^{-1}	NA
Example 2a (N=8)	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	2.4706×10^{-4}	NA
Example 2b (N=8)	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.7956×10^{-8}	6.6737×10^{-6}	NA

NA: not available (always applies to the best approach)

(N=8), the P-values obtained using the IDE and the SADE algorithms were larger than 0.05, suggesting that their optimization results were comparable to those of the RDE algorithm for that example. Therefore, the results from Table 4 verify that the RDE algorithm is the best among the six approaches, and that its results were statistically different from those of the other five approaches for all problems except Example 1b (N=8).

Fig. 9 displays the MSEs (in logarithmic scale) of the desired and estimated signals obtained by the six algorithms for six problems. Clearly, both the PSO and IPSO algorithms have larger convergence rates than the other four approaches at the beginning of the optimization process. However, the two PSO algorithms become trapped early in the local optimal solutions,

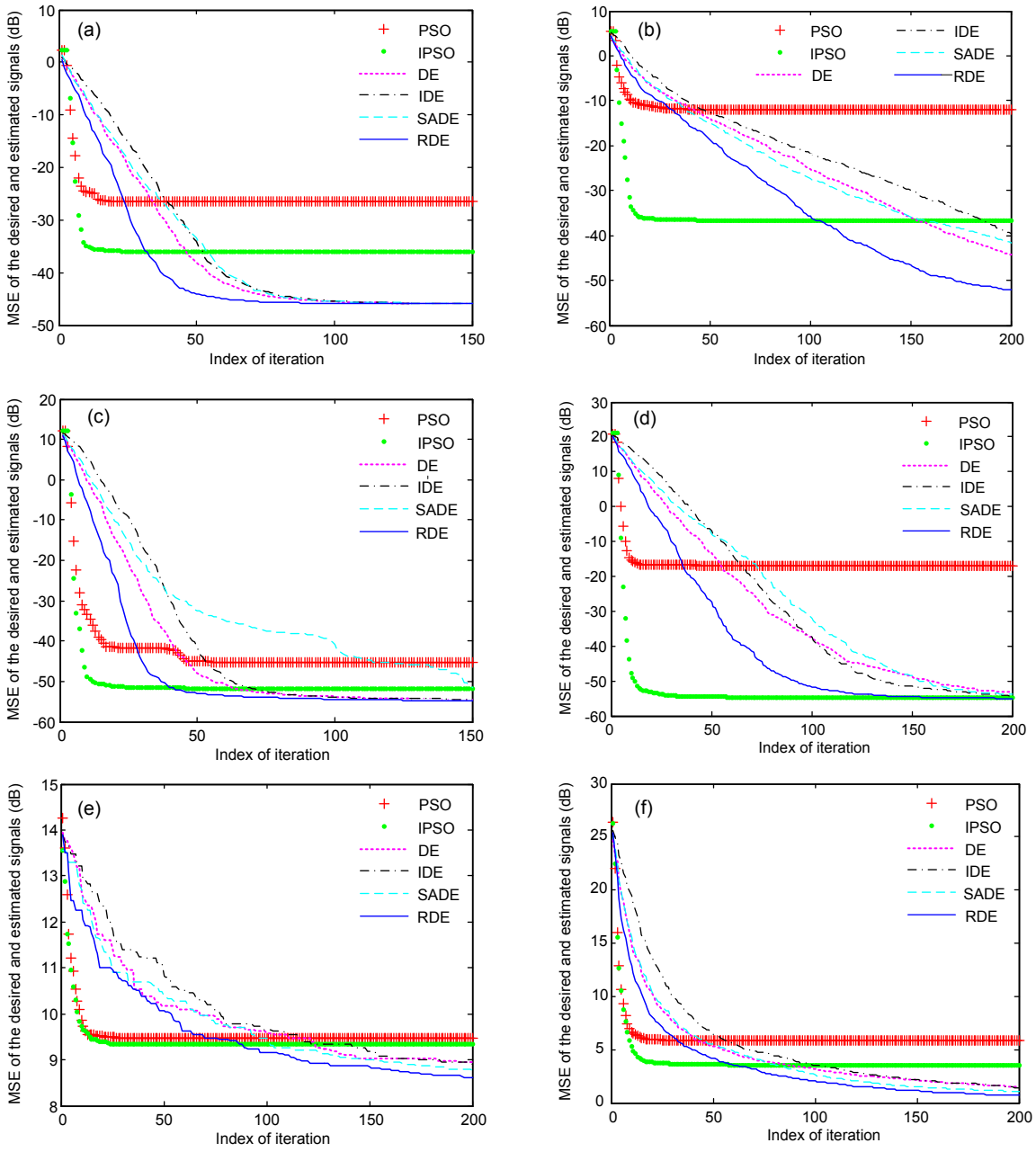

Fig. 9 The MSEs obtained by using six approaches for solving six problems

(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)

which can be considered as premature convergence. In contrast, the four DE algorithms converge gradually to the global optimal solutions. Furthermore, the RDE algorithm exhibits better convergence than the other three DE algorithms. In Example 1a (N=8), Example 2a (N=8), and Example 2b (N=8), the RDE algorithm reached the lowest levels. With respect to the other three problems, the four DE algorithms converged to comparable levels. In fact, the RDE algorithm was still superior to the other three DE algorithms in solving these three problems (Table 3).

6 Conclusions

In this paper, we propose a ranked differential evolution (RDE) algorithm for identifying nonlinear discrete-time systems based on a truncated second-order Volterra model. The kernels of the Volterra model are optimized by the RDE algorithm so that the Volterra filter output is very close to the actual system output. In addition, we investigated the effects of different memory sizes on modeling performance. Experimental results indicate that the RDE algorithm performs well for nonlinear system identification based on the truncated second-order Volterra model. Moreover, in most cases it can find better objective function values (or minimum mean square errors) than the other five evolutionary algorithms tested.

References

Babu, B.V., Angira, R., 2006. Modified differential evolution (MDE) for optimization of non-linear chemical processes. *Comput. Chem. Eng.*, **30**(6-7):989-1002. [doi:10.1016/j.compchemeng.2005.12.020]

Brest, J., Greiner, S., Boskovic, B., *et al*., 2006. Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.*, **10**(6):646-657. [doi:10.1109/TEVC.2006.872133]

Chang, W.D., 2012. Volterra filter modeling of nonlinear discrete-time system using improved particle swarm optimization. *Dig. Signal Process.*, **22**(6):1056-1062. [doi:10.1016/j.dsp.2012.07.005]

Cheng, C.H., Powers, E.J., 2001. Optimal Volterra kernel estimation algorithms for a nonlinear communication system for PSK and QAM inputs. *IEEE Trans. Signal Process.*, **49**(1):147-163. [doi:10.1109/78.890357]

Contan, C., Kirei, B.S., Topa, M.D., 2013. Modified NLMF adaptation of Volterra filters used for nonlinear acoustic echo cancellation. *Signal Process.*, **93**(5):1152-1161. [doi:10.1016/j.sigpro.2012.11.017]

Derrac, J., García, S., Molina, D., *et al*., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.*, **1**(1):3-18. [doi:10.1016/j.swevo.2011.02.002]

Ji, W., Gan, W.S., 2012. Identification of a parametric loudspeaker system using an adaptive Volterra filter. *Appl. Acoust.*, **73**(12):1251-1262. [doi:10.1016/j.apacoust.2012.03.007]

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks, p.1942-1948. [doi:10.1109/ICNN.1995.488968]

Krall, C., Witrisal, K., Leus, G., *et al*., 2008. Minimum mean-square error equalization for second-order Volterra systems. *IEEE Trans. Signal Process.*, **56**(10):4729-4737. [doi:10.1109/TSP.2008.928167]

Kuruoğlu, E.E., 2002. Nonlinear least l_p-norm filters for nonlinear autoregressive α-stable processes. *Dig. Signal Process.*, **12**(1):119-142. [doi:10.1006/dspr.2001.0416]

Li, X., Yin, M., 2012. Optimal synthesis of linear antenna array with composite differential evolution algorithm. *Sci. Iran.*, **19**(6):1780-1787. [doi:10.1016/j.scient.2012.03.010]

Mleczko, M., Postema, M., Schmitz, G., 2009. Discussion of the application of finite Volterra series for the modeling of the oscillation behavior of ultrasound contrast agents. *Appl. Acoust.*, **70**(10):1363-1369. [doi:10.1016/j.apacoust.2008.09.012]

Nam, S.W., Powers, E.J., 2003. Volterra series representation of time-frequency distributions. *IEEE Trans. Signal Process.*, **51**(6):1532-1537. [doi:10.1109/TSP.2003.811241]

Shi, Y.H., Eberhart, R.C., 1999. Empirical study of particle swarm optimization. Proc. Congress on Evolutionary Computation, p.1945-1950. [doi:10.1109/CEC.1999.785511]

Storn, R., Price, K., 1995. Differential Evolution—a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. International Computer Science Institute, Berkeley, USA.

Sumar, R.R., Coelho, A.A.R., Coelho, L.D.S., 2010. Computational intelligence approach to PID controller design using the universal model. *Inform. Sci.*, **180**(20):3980-3991. [doi:10.1016/j.ins.2010.06.026]

Tang, H., Liao, Y.H., Cao, J.Y., *et al*., 2010. Fault diagnosis approach based on Volterra models. *Mech. Syst. Signal Process.*, **24**(4):1099-1113. [doi:10.1016/j.ymssp.2009.09.001]

Wilcoxon, F., 1945. Individual comparisons by ranking methods. *Biometr. Bull.*, **1**(6):80-83.

Zhang, J.S., Zhao, H.Q., 2010. A novel adaptive bilinear filter based on pipelined architecture. *Dig. Signal Process.*, **20**(1):23-38. [doi:10.1016/j.dsp.2009.06.006]

Zou, D.X., Liu, H.K., Gao, L.Q., *et al*., 2011. An improved differential evolution algorithm for the task assignment problem. *Eng. Appl. Artif. Intell.*, **24**(4):616-624. [doi:10.1016/j.engappai.2010.12.002]