International Journal of Automation and Computing 6(2), May 2009, 137-144
DOI: 10.1007/s11633-009-0137-0

An Improved Differential Evolution Trained Neural Network Scheme for Nonlinear System Identification

Bidyadhar Subudhi* Debashisha Jena

Center for Industrial Electronics & Robotics, Department of Electrical Engineering, National Institute of Technology, Rourkela 769008, India

Abstract: This paper presents an improved nonlinear system identification scheme using differential evolution (DE), neural network (NN) and Levenberg Marquardt algorithm (LM). With a view to achieve better convergence of NN weights optimization during the training, the DE and LM are used in a combined framework to train the NN. We present the convergence analysis of the DE and demonstrate the efficacy of the proposed improved system identification algorithm by exploiting the combined DE and LM training of the NN and suitably implementing it together with other system identification methods, namely NN and DE+NN on a number of examples including a practical case study. The identification results obtained through a series of simulation studies of these methods on different nonlinear systems demonstrate that the proposed DE and LM trained NN approach to nonlinear system identification can yield better identification results in terms of time of convergence and less identification error.

Keywords: Differential evolution, neural network (NN), nonlinear system identification, Levenberg Marquardt algorithm

1 Introduction

There is an increasing interest in nonlinear system identification research from the viewpoint of industry and real world applications. A number of nonlinear system identification techniques have been proposed, such as the Voltera series, Winner-Hammerstein model and polynomial identification[1] methods which involve computational complexities. Further, there is a lot of research directed towards applying NNs[2, 3] for nonlinear system identification due to its function approximation capabilities. Multi-layered perceptrons (MLP) with back-propagation training[3] have been successfully employed for system identification of a number of complex dynamical systems. Although the NN has been proved to be a successful technique for nonlinear system identification, there is still little concern about its convergence and problem of being trapped at local minima. In addition to the NNs, the wavelet network techniques[2] have also been applied to system identification of nonlinear systems in which adaptive techniques such as back propagation algorithm are found to provide better accuracy compared to non-adaptive ones such as the Voltera series, Winner-Hammerstein modeling and polynomial methods.

During the last three decades evolutionary algorithms (EAs), such as genetic algorithm (GA), and evolutionary strategies (ES) have become very popular as function optimizers, because they are easy to implement and exhibit fair performance for a wide range of functions. However, continued development in the research community has observed that the tuning of the GA parameters in complex search spaces is difficult.

The differential evolution (DE) is a population based stochastic optimization method similar to GA that has found an increasing interest in recent years as an optimization technique due to its achievement of a global minimum[4, 5]. DE is an effective, efficient, and robust optimization method capable of handling nonlinear and multimodal objective functions. The beauty of DE is its simple and compact structure which uses a stochastic direct search approach and utilizes common concepts of EAs. Furthermore, DE uses a few easily chosen parameters and provides excellent results for a wide set of benchmark and real-world problems. Experimental results have shown that DE has good convergence properties and outperforms other well-known EAs[5]. Therefore, there is a wide scope for using DE approach to neural weight optimization.

Although an evolutionary computing technique such as the GA has been combined to the NN for nonlinear system identification[6], it is not self-adaptive in nature. As a result, it is more difficult to tune its step size in the successive generations. Hence, in the case of GA-NN, fast convergence is hard to achieve. In this paper, we attempt to exploit the advantages of DE which requires a lesser number of parameters to tune and is easy to code and is self-adaptive in nature because in the differential evolution the differential term is very large at the moment of starting. As the solution approaches to the global minimum, the differential term automatically changes to a low value. As the differential term changes automatically with the progress of the search process, the algorithm becomes self-adaptive. This will be explained in more detail in Section 2.

It may be noted that using the DE approach alone may yield a slow convergence speed although it provides the global minimum as compared to classical optimization methods such as the gradient descent and Levenberg Marquardt (LM) methods. Therefore, we propose a DE+LM+NN approach in view of achieving the global minimum with a good convergence speed. In this paper, a differential evolution method combined with LM has been applied as a global optimization method for training a feed-forward NN. In the proposed scheme, the DE is used to train the NN that is chosen as a suitable candidate for nonlinear system identification. After observing the trends of training towards a minimum through DE, the network is

Manuscript received December 19, 2007; revised November 28, 2008
*Corresponding author.
E-mail address: bidyadharnitrkl@gmail.com

then trained by LM. The role of the DE here is to approach the global minimum point and then LM is used to move forward to achieve a fast convergence. The nonlinear systems considered in [7, 8] have been chosen in this paper for demonstrating the efficacy of the proposed system identification approach.

The rest of the paper is organized as follows. Section 2 includes a brief review on differential evolution. In Section 3, we discuss the convergence analysis of the DE. Section 4 gives an overview of the proposed DE+LM+NN algorithm. In Section 5, results are included and discussed to verify the effectiveness of the proposed method. Finally, concluding remarks on the improved identification scheme are presented in Section 6.

2 Review on differential evolution

In a population of potential solutions to an optimization problem within an n-dimensional search space, a fixed number of vectors are randomly initialized, then evolve over time to explore the search space and to locate the minima of the objective function.

In DE, individuals are represented as real-valued vectors. For each generation of the evolution process, each individual (target individual) of the population competes against a new individual (trial individual) for survival to the next generation. In the next generation, the fitter of the two individual survives. The trial individual is created by recombining the target individual with another individual created by mutation (mutant individual). Mutation is performed on the best individual found so far in the evolution process. For each target vector $x_{i,G}$, a mutant vector is produced using the following formula

$$\begin{aligned} v_{i,G+1} &= x_{r1,G} + Fz_2 \\ z_2 &= (x_{r2,G} - x_{r3,G}) \end{aligned} \tag{1}$$

where $i, r_1, r_2, r_3 \in \{1, 2, \cdots, NP\}$ are randomly chosen and must be different from each other. Considering (1), it can be seen that the pdf of the differential population z_2 used during the mutation changes automatically as the generation proceeds and eventually the solution converges towards the global minimum.

Let P be the number of populations. In Fig. 1, five populations X_1, $X2$, $\cdots$, X_5 produce ten numbers of vector differences in one direction and twenty numbers in both directions as shown in Fig. 2. Similarly, for P populations there will be $P(P-1)/2$ vector differences in one direction and $P(P-1)$ in both directions. This implies that the mean of the pdf is always zero and the shape of the distribution changes automatically in successive generations depending on the surface of the objective function being searched as shown in Fig. 3.

In (1), F is the mutation factor. Recombination creates an offspring (trial individual) by selecting parameters from either the target individual or the mutant individual. There are two methods of recombination in DE, namely, binomial recombination and exponential recombination. In the binomial recombination, a series of binomial experiments are conducted to determine which parent contributes which parameter to the offspring. Each experiment is mediated by

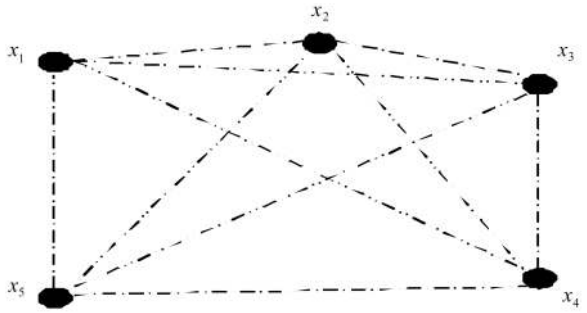

Fig. 1 Five populations

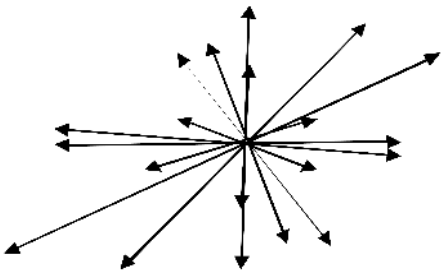

Fig. 2 Twenty vector differences

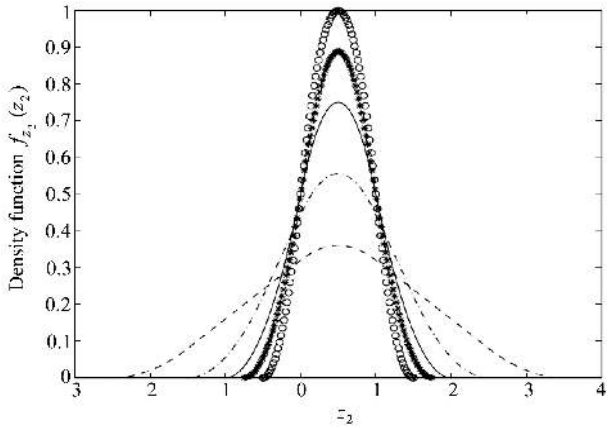

Fig. 3 Probability density functions

a crossover constant, CR, $(0 \leqslant CR \leqslant 1)$. Starting at a randomly selected parameter, the source of each parameter is determined by comparing CR to a uniformly distributed random number from the interval [0, 1). If the random number is greater than CR, the offspring gets its parameter from the target individual; otherwise, the parameter comes from the mutant individual. In exponential recombination, a single contiguous block of parameters of random size and location is copied from the mutant individual to a copy of the target individual to produce an offspring. A vector of solutions are selected randomly from the mutant individuals when $rand_j$ ($rand_j \in [0, 1]$ is a random number) is less than CR.

In crossover, the parent vector is mixed with the mutated vector to produce a trial vector $t_{ji,t+1}$

$$t_{ji,t+1} = \begin{cases} v_{ji,t+1}, & \text{if } rand_j \leqslant CR \\ w_{ji,t+1}, & \text{if } rand_j > CR \end{cases}, \quad j = 1, 2, \cdots, D \tag{2}$$

where D is the number of parameters to be optimized.

At each generation, new vectors are generated by the combination of vectors randomly chosen from the current population (mutation). The upcoming vectors are then mixed with a predetermined target vector. This operation is called recombination and produces the trial vector. Finally, the trial vector is accepted for the next generation iff it yields a reduction in the value of the objective function. This last operator is referred to as a selection.

Fig. 4 shows a two-dimensional objective function illustrating different vectors x_i which are important in differential evolution. It shows the process of generating a trial vector for the scheme explained in (2).

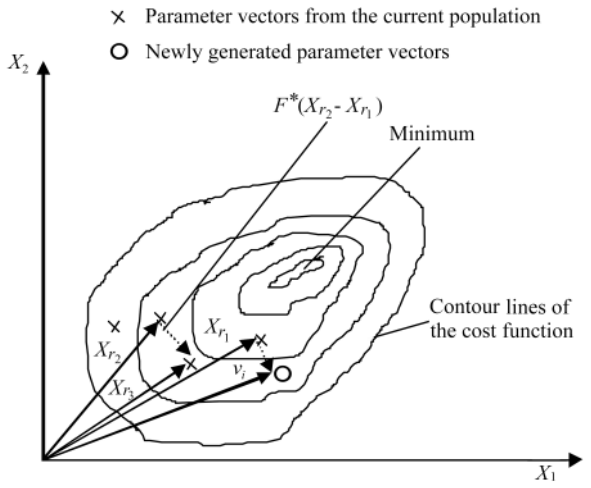

Fig. 4 A two-dimensional objective function

3 Convergence analysis of DE

Let $\boldsymbol{P}_G = (\boldsymbol{w}_{1,G}, \cdots, \boldsymbol{w}_{NP,G})^{\mathrm{T}}$, $G = 1, \cdots, G_{\max}$ (where $G_{\max}$ is the maximum number of generations) be the random population of size NP at step $G \geqslant 0$, and $F_G = \min\{f(\boldsymbol{P}_{G,i}) : i = 1, \cdots, NP\}$ be the best fitness value within the population at step $G \geqslant 0$. As soon as the random variable F_G attains the value of the global maximum, f^*, it is ensured that the population contains an individual representing the global solution of the minimization problem. Ideally, this event should happen after a finite number of steps with probability one regardless of the initialization of the DE algorithm.

Property 1. In DE, it is known that the best solutions found in the process of evolution in the current generation are carried over to the next generation. This property guarantees that the global optimum will be found in finite time and never be lost once it is achieved. Thus, the property above shows that the random sequence $(F_G : G \geqslant 0)$ converges to the limit f^*.

Let $(w_1, w_2, \cdots, w_n) \in P^n$ denote the population of parents known as target individuals. An offspring is produced as follows. At first m number of parents are selected to serve as mates for the mutation process. This operation is denoted as follows

$$mat : \boldsymbol{P}^n \to \boldsymbol{P}^m \quad \text{where } 3 \leqslant m \leqslant n.$$

These individuals are then mutated by the following procedure

$$mut : \boldsymbol{P}^m \to \boldsymbol{P}.$$

Thus, it yields a mutated individual. Finally, recombination with the target individual, we have a trial individual using the following process

$$reco : \boldsymbol{P} \to \boldsymbol{P}.$$

Now, the selection is done by the trial and target individuals through a selection procedure given as

$$sel : \boldsymbol{P} \to \boldsymbol{P}.$$

The above selection procedure decides which ones will serve as the new parents in the next generation. Thus, a single generation of differential evolution can be described as follows.

$$\forall i \in \{1, \cdots, NP\} : \; u_i = reco(mut(mat(w_1 \cdots w_n)))$$

$$\forall i \in \{1, \cdots, NP\} : \; y_i = \{sel(w_i, y_i)\}.$$

After this operational description, differential evolution is in the position of defining some assumptions about the properties of the variation and selection operators.

Assumption 1. Every parent may be selected for mating and can be changed to an arbitrary other individual by a finite number of successive mutations, i.e., for every $w \in \boldsymbol{P}$ there exists a finite path such that $\Pr\{w_{i+1} = mut(w_i)\} = 1$ for $i = 1, \cdots, (G-1)$.

Assumption 2. Every mutant individual is altered by the recombination with the minimum probability $p_{cr} > 0$.

$$\forall i \in \{1, \cdots, NP\} :$$
$$\Pr\{u_i = reco(mut(mat(w_1 \cdots w_n)))\} \geqslant p_{cr} > 0.$$

Assumption 3. Every trial individual competing for survival will survive with a minimum probability of 0.5.

$$\forall i \in \{1, \cdots, NP\} : \; \Pr\{sel(w_i, y_i)\} = 0.5.$$

Assumption 4. The best individual among the competitors in the selection process will survive with probability 1.

Theorem 1. (Convergence analysis of the DE algorithm) If the Assumptions 1–3 are valid, then the differential evolution visits the global optimum after a finite number of generations with probability one regardless of the initialization.

Proof. Let random variable $T = \{G \geqslant 0 : F_G = f^*\}$ denote the first hitting time of the global solution. An evolutionary algorithm is said to visit the global optimum in finite time with probability one if $\Pr\{T < \infty\} = 1$ regardless of the initialization.

Let $\boldsymbol{P}^* = \{w \in \boldsymbol{P} : f(w) = f^*\}$ be the set of globally optimal solutions. Owing to Assumption 1, there exists a finite path from an arbitrary $w \notin \boldsymbol{P}^*$ to some $w^* \in \boldsymbol{P}^*$ that can be traversed by successive mutations. Let t_x be the length of the path between $w \notin \boldsymbol{P}^*$ to the set $w^* \in \boldsymbol{P}^*$.

Now, we consider an arbitrary parent w of some population known as the target individual. Assumption 1 ensures that this parent passes the mutation process with every change with probability one. The probability that the mutated individual transits to the next point of the path towards $w^* \in \boldsymbol{P}^*$ by recombination is guaranteed to be at least $p_{cr} > 0$ by Assumption 2.

Owing to Assumption 3, this offspring will survive the selection process at least with probability $p_s = 0.5$. Thus, the probability that parent $w \notin \boldsymbol{P}^*$ transits to a parent representing the next point on the path to $w^* \in \boldsymbol{P}^*$ is at least $0.5\,p_{cr} > 0$. Consequently the probability that a globally optimal solution has not been found is $(1 - 0.5\,p_{cr})$.

A G_w fold repetition of this argumentation shows that the probability for a globally optimal solution not to be found after G_w generations is at most $(1 - 0.5\,p_{cr})^{G_w}$ which converges exponentially fast to zero as $G_w \to \infty$. This immediately implies $\Pr\{T < \infty\} = 1$, where $T = \{G \geqslant 0 : F_G = f^*\}$. Thus, a global optimum will be visited for the first time after a finite number of iterations with probability one. □

4 Proposed DE+LM+NN system identification scheme training algorithm

In this section, we describe how a DE is applied to the training NN in the framework of system identification (see the following Algorithm 1). According to Step 7, in the Algorithm 1, after reaching a particular value of ε, the algorithm is switched from the global search space of the differential evolution algorithm to a local search by exploiting the LM technique. In differential evolution, at the moment of starting, the differential term is very large. As the solution approaches global minimum, the differential term automatically changes to a low value. Thus, during the initial period of the search process, the convergence speed is fast and the search space is very large. However, during the latter stage of the search process due to the small differential term, $(x_{r2,G} - x_{r3,G})$, the algorithm becomes slow which may take more time to converge. To overcome such a situation, we then exploit the LM, a gradient based algorithm, to take over the optimization to achieve the fast convergence. The role of LM at this moment is to increase the convergence speed for reaching the global minimum. Our objective here is to apply DE to the weight optimization of a typical feed-forward NN. The output of a feed-forward NN is a function of synaptic weights $\boldsymbol{w}$ and input values $\boldsymbol{x}$, i.e., $\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{w})$. In the training process, both the input vector $\boldsymbol{x}$ and the output vector $\boldsymbol{y}$ are known, and the synaptic weights in $\boldsymbol{w}$ are adapted to obtain appropriate functional mappings from the input $\boldsymbol{x}$ to the output $\boldsymbol{y}$. Generally, the adaptation can be carried out by minimizing the network error function E which is of the form $\mathrm{E}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{w}))$. In this work, we have taken E as mean squared error, i.e., $\mathrm{E} = [\sum_{k=1}^{N} [\boldsymbol{y} - f(\boldsymbol{x}, \boldsymbol{w})]^2]/N$, where N is the number of data considered.

The optimization goal is to minimize the objective function E by optimizing the values of the network weights $\boldsymbol{w}$, where $\boldsymbol{w} = (w_1, \cdots, w_d)$.

Algorithm 1. (DE+LM+NN identification algorithm)

Step 1. Initialize population.

$$\boldsymbol{P}_G = (\boldsymbol{w}_{1,G}, \cdots, \boldsymbol{w}_{NP,G})^{\mathrm{T}}, \quad G = 1, \cdots, G_{\max}$$

$$\boldsymbol{w}_{i,G} = (w_{1,i,G}, \cdots, w_{D,i,G}), \quad i = 1, \cdots, NP$$

where D is the number of weights in the weight vector and in $\boldsymbol{w}_{i,G}$, i is index to the population and G is the generation to which the population belongs.

Step 2. Evaluate all the candidate solutions inside population for a specified number of iterations.

Step 3. For each i-th candidate in the population, selects the random variables r_1, r_2, $r_3 \in \{1, 2, \cdots, NP\}$.

Step 4. Apply mutation operator to yield a mutant vector, i.e.,

$$v_{j,i,G+1} = w_{j,r1,G} + F(w_{j,r2,G} - w_{j,r3,G}), \text{ for } j = 1, \cdots, D \ (i \neq r_1 \neq r_2 \neq r_3) \in \{1, \cdots, NP\} \text{ and } F \in (0, 1]$$

Step 5. Apply crossover to produce trial vector.

$$t_{j,i,G+1} = \begin{cases} v_{j,i,G+1}, & \text{if } rand_j[0,1) \leqslant CR \\ w_{j,i,j}, & \text{otherwise} \end{cases}$$

where $CR \in [0, 1]$.

Step 6. Apply selection between the trial vector and target vector.

$$\boldsymbol{w}_{i,G+1} = \begin{cases} t_{i,G+1}, & \text{if } \mathrm{E}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{w}_{i,G+1})) \leqslant \mathrm{E}(\boldsymbol{y}, f(\boldsymbol{x}, \boldsymbol{w}_{i,G})) \\ \boldsymbol{w}_{i,G}, & \text{otherwise} \end{cases}$$

Step 7. If $\mathrm{E} \leqslant \varepsilon$, where $\varepsilon > 0$ then go to Step 8.

Step 8. Initialize the weight matrix of Levenberg-Marquardt algorithm. Find out the value of E.

Step 9. Compute the Jacobian matrix $\boldsymbol{J}(w)$.

Step 10. Find Δw using the following equation

$$\Delta w = \left[\boldsymbol{J}^{\mathrm{T}}(w)\, J(w) + \mu I\right]^{-1} \boldsymbol{J}^{\mathrm{T}}(w)\, \mathrm{E}.$$

Step 11. Recompute E using $(w + \Delta w)$. If this new E is smaller than that computed in Step 7, then reduce μ and go to Step 1, where μ is the damping factor.

Step 12. The algorithm is assumed to have converged when the norm of the gradient, i.e., $\|\nabla \mathrm{E}\| = \left\|\boldsymbol{J}^{\mathrm{T}}(w)\boldsymbol{y} - f(\boldsymbol{x}, w)\right\|$ is less than some predetermined value, or when the sum of squares of errors has been reduced to some error goal.

5 Results and discussion

We present here the performance achieved through using the proposed DE+LM+NN scheme to a number of benchmark problems as follows.

Example 1. The nonlinear system[5] to be identified is expressed by

$$y_p(k+1) = \frac{y_p(k)[y_p(k-1)+2][y_p(k)+2.5]}{8.5 + [y_p(k)]^2 + [y_p(k-1)]^2} + u(k) \quad (3)$$

where $y_p(k)$ is the output of the system at the k-th time step and $u(k)$ is the plant input which is a uniformly bounded function of time. The plant is stable at $u(k) \in [-2\ 2]$. The identification model is in the form of

$$y_{pi}(k+1) = N(y_p(k), y_p(k-1)) + u(k) \quad (4)$$

where $N(y_p(k), y_p(k-1))$ is the nonlinear function of $y_p(k)$ and $y_p(k-1)$. The inputs to the NN are $y_p(k)$ and $y_p(k-1)$. The output from the NN is $y_{pi}(k+1)$. The goal is to train the NNs such that when an input $u(k)$ is presented to the NN and to the nonlinear system, the NN outputs $y_{pi}(k)$ and the actual nonlinear system output $y_p(k)$ will closely match.

1) **NN identifier**

The NN identifier structure consisted of eleven neurons in the hidden layer. After 1000 epochs the training of the neural identifier stopped. After the training, its prediction capability was tested for input.

$$u(k) = \begin{cases} 2\cos(\frac{2\pi k}{100}), & \text{if } k \leqslant 200 \\ 1.2\sin(\frac{2\pi k}{20}), & \text{if } 200 < k \leqslant 500. \end{cases}$$

Fig. 5 shows the system identification results obtained using NN. The error is more at time steps 100 and 200. Fig. 6 shows the identification error.

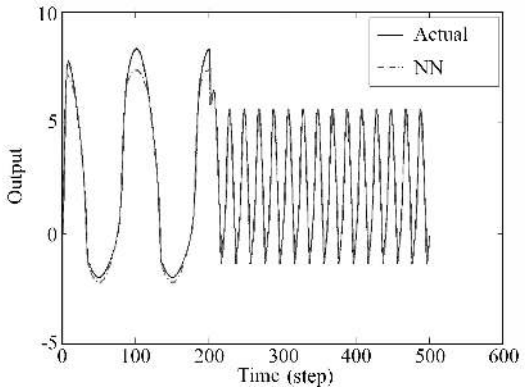

Fig. 5 Identified and actual models (NN identifier)

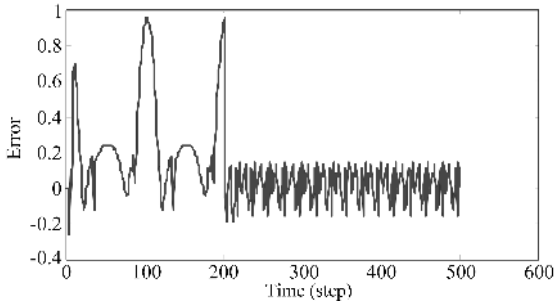

Fig. 6 Error in modeling (NN identifier)

2) **DE+NN identifier**

Fig. 7 shows the identification performance of the system using NN and differential evolution. Here, the network was trained by using differential evolution instead of classical ones such as gradient descent and LM algorithms. The results obtained with DE+NN indicate no significant improvement over the previously discussed NN identifier. Here, eleven hidden layer neurons and thousands of epochs were also taken. Fig. 8 shows the identification error in the case of the DE+NN approach.

3) **DE+NN+LM identifier**

Fig. 9 gives the result of proposed DE+LM+NN scheme. In this case the network was trained by both DE and LM algorithms. Here, eleven hidden layer neurons were considered. Fig. 10 shows its identification error curve for DE+LM+NN system identification. From Fig. 10, it is clear that identification error is smaller than errors in NN and NN+DE. This result clearly indicates that accurate identification of a nonlinear system is achieved, i.e., the superior identification capability of the proposed scheme over the other NN and DE+NN methods.

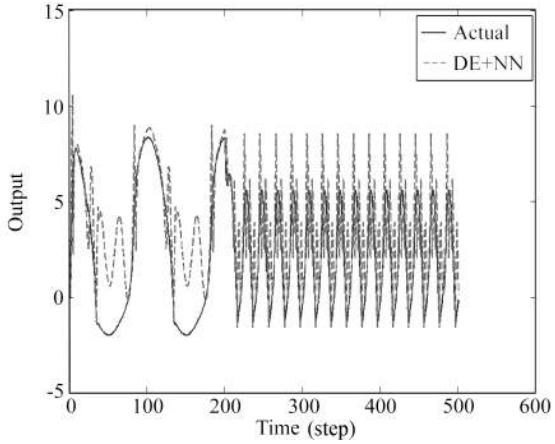

Fig. 7 Identified and actual models (DE+NN identifier)

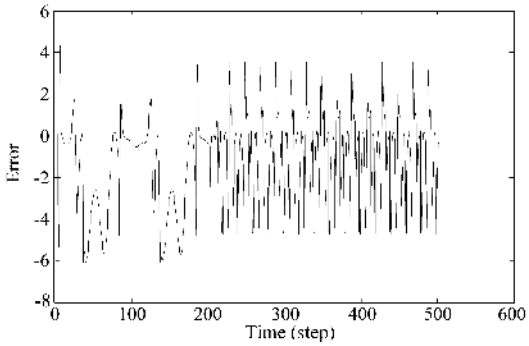

Fig. 8 Error in modeling (DE+NN identifier)

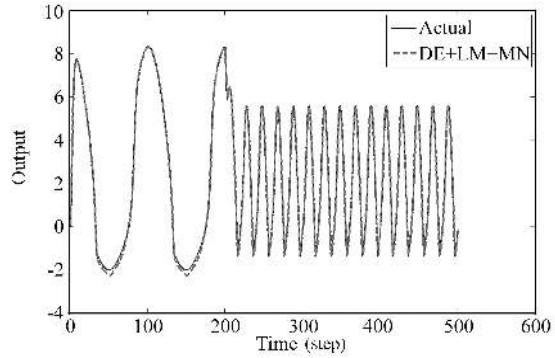

Fig. 9 Identified and actual models (DE+LM+NN identifier)

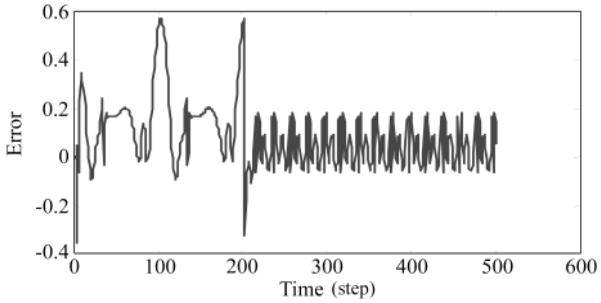

Fig. 10 Error in modeling (DE + LM + NN identifier)

Example 2. The plant[7] to be identified is governed by

$$y_p(k+1) = 0.3\,y_p(k) + 0.6\,y_p(k-1) + N\,[u(k)] \quad (5)$$

where the unknown function has the form

$$N(u) = 0.6\,\sin(\pi\,u) + 0.3\,\sin(3\pi\,u) + 0.1\,\sin(5\pi\,u). \quad (6)$$

In order to identify the plant, a series parallel model governed by the difference equation

$$\hat{y}_p(k+1) = 0.3\,y_p(k) + 0.6\,y_p(k-1) + N\,[u(k)] \quad (7)$$

was used.

1) **NN identifier**

The NN identifier structure consisted of eleven neurons in the hidden layer. After 1500 epochs, the training of the neural identifier stopped. After the training, its prediction capability was tested for input given as

$$u(k) = \sin\frac{2\pi k}{250}.$$

As shown in Fig. 11, the learned network predicted the nonlinear system outputs. Fig. 12 shows the identification error. The figures clearly indicate the poor identification performance of the neural identifier.

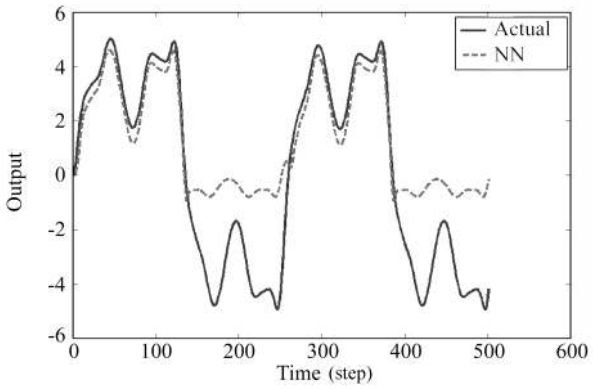

Fig. 11 Identified and actual models (NN identifier)

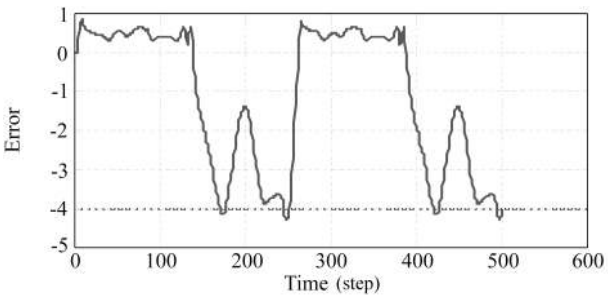

Fig. 12 Error in modeling (NN identifier)

2) **DE+NN identifier**

Fig. 13 shows the identification performance of the system using differential evolution. Here, the network was trained by using differential evolution instead of classical ones such as gradient descent and LM algorithms. The results obtained with DE+NN indicate no significant improvement over the previously discussed existing ones.

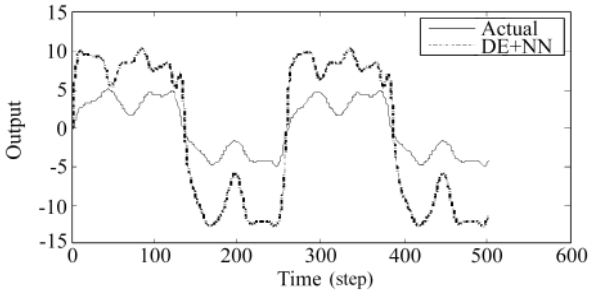

Fig. 13 Identified and actual models (DE identifier)

3) **DE+LM+NN identifier**

Fig. 14 gives the result of proposed DE+LM+NN scheme. In this case, the network was trained both by DE and LM algorithms. This result clearly indicates the superior identification capability of the proposed scheme over the other two methods discussed, i.e., the NN and DE+NN approaches. Fig. 15 shows its identification error curve for DE+LM+NN system identification.

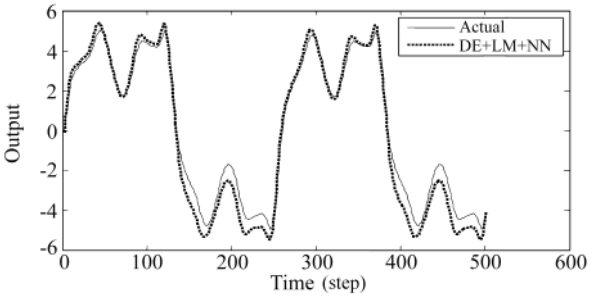

Fig. 14 Identified and actual models (DE+LM+NN identifier)

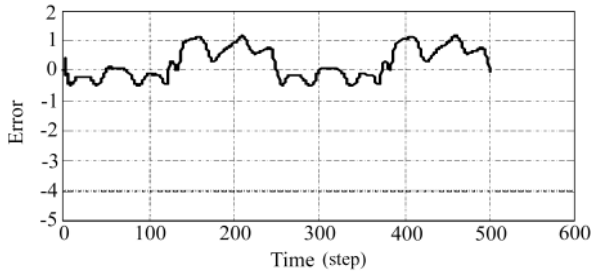

Fig. 15 Error in modeling (DE+LM+NN identifier)

Example 3. The Box and Jenkins′ gas furnace data are frequently used in performance evaluation of system identification methods[8]. The data can be obtained from the site. The example consists of 296 input–output samples recorded with a sampling period of 9 s. The gas combustion process has one variable, the gas flow $u(k)$, and one output variable, the concentration of carbon dioxide (CO_2) $y(k)$. The instantaneous values of output $y(k)$ have been regarded as being influenced by ten variables $y(k-1)$, $y(k-2)$, $y(k-3)$, $y(k-4)$, $y(k-5)$, $u(k-1)$, $u(k-2)$, $u(k-3)$, $u(k-4)$, and $u(k-5)$. In the literature, the number of variables influencing the output varies from 2 to 10. In the proposed method, ten variables were chosen. The results shown gives a comparison of the identification methods such as NN trained with conventional methods and NN trained with DE and hybrid differential evolution methods. For all the methods,

eleven hidden layer neurons were taken and the results were obtained after 1000 epochs. The number of training data was taken as 100 for all the cases and the other 196 data were the test data.

1) **NN identifier**

Fig. 16 shows the graphs of the identified results obtained with NN and the actual system. Here, the NN fails to identify the system dynamics at iteration of 260 thus leading to a big identification error.

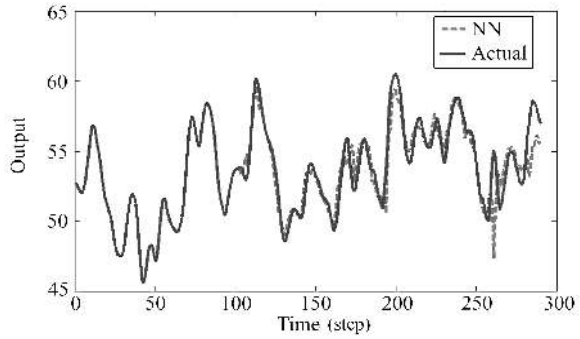

Fig. 16 Identified and actual models (NN identifier)

2) **DE+NN identifier**

The DE+NN identified system dynamics and the actual system dynamics were plotted in Fig. 17. It is observed that there is no improvement in identification with respect to the previous one, i.e., the NN identifier.

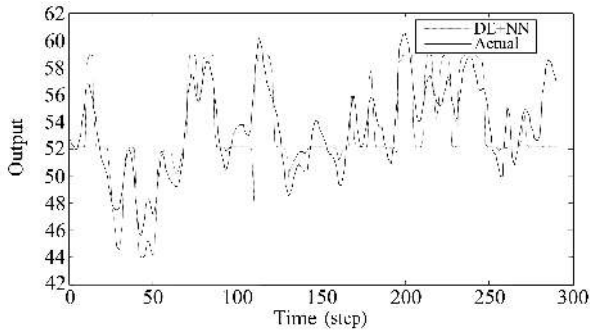

Fig. 17 Identified and actual models (DE+NN identifier)

3) **DE+LM +NN identifier**

The DE+LM+NN identified system dynamics and the actual system dynamics were plotted in Fig. 18.

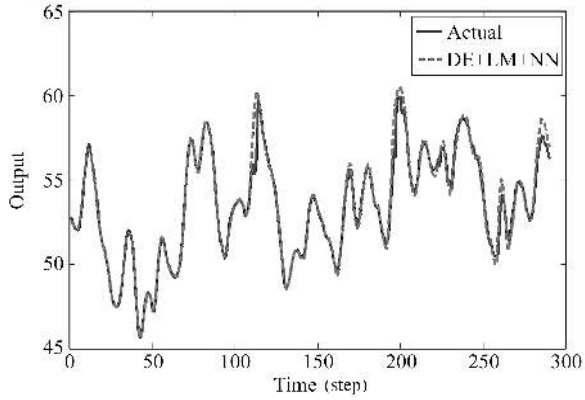

Fig. 18 Identified and actual models (DE+LM+NN identifier)

Fig. 19 gives the comparison of errors in modeling between the proposed DE+LM+NN identifier and the NN identifier. In this case, the network was trained both by DE and LM algorithm. From Fig. 18, we see that the proposed DE+LM+NN scheme has exhibited the expected identification performance, i.e., the error between the true system and the identified one is the minimum. Table 1 summaries the performance of the proposed method of system identification (DE+LM+NN) over the existing ones (NN, DE+NN) for different examples and case studies.

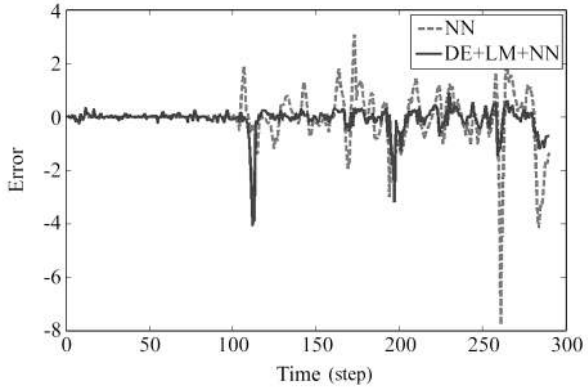

Fig. 19 Comparison of errors in modeling (NN vs (DE + LM + NN identifier))

Table 1 Comparison of performances of three methods

Method	Time of convergence (s)	Mean squared error (MSE)	Example
Only NN	17.490	0.0047	
DE+NN	50.112	6.8830	Example 1
DE+LM+NN	24.899	0.0030	
Only NN	18.991	0.3115	
DE+NN	121.148	3.5470	Example 2
DE+LM+NN	30.985	0.0059	
Only NN	20.929	0.0038	
DE+NN	102.347	2.6680	Example 3
DE+LM+NN	54.895	0.0001	

6 Conclusions

In this paper, we have clearly described the scope of improving the nonlinear system identification strategy by the improved training of the NNs using both the differential evolution and LM algorithms. In the proposed identification framework, differential evolution is used only to find approximate values in the vicinity of the global minimum. These approximate weight values are then used as starting values for a fast convergence algorithm, i.e., LM algorithm. From the results presented in Section 5, it is clear that there is certainly an improvement in identification performance for nonlinear systems over the existing approaches. In comparison with the DE+NN approach, the proposed DE+LM+NN approach provides better system identification performance in terms of speed of convergence and identification capability. Although good identification results have been obtained in this work, it seems that an extensive research is still required to implement different concepts of

the DE, such as the opposition based differential evolution (ODE) in system identification and to verify whether the proposed learning scheme can bring clear improvement in terms of convergence speed and better identification accuracy.

Acknowledgement

The authors acknowledge their sincere thanks to the anonymous reviewers and the editor for suggesting a number of improvements for enhancing the quality of the paper.

References

[1] S. Chen, S. A. Billings, W. Luo. Orthogonal Least Squares Methods and Their Application to Non-linear System Identification. *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[2] S. A. Billings, H. L. Wei. A New Class of Wavelet Networks for Nonlinear System Identification. *IEEE Transactions on Neural Networks*, vol. 16, no. 4, pp. 862–874, 2005.

[3] K. S. Narendra, K. Parthaasarathy. Identification and Control of Dynamical Systems Using Neural Networks. *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[4] R. Storn. System Design by Constraint Adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 1, pp. 22–34, 1999.

[5] J. Ilonen, J. K. Kamarainen, J. Lampinen. Differential Evolution Training Algorithm for Feed Forward Neural Networks. *Neural Processing Letters*, vol. 17, no. 1, pp. 93–105, 2003.

[6] O. Ludwig Jr., P. C. Gonzalez, A. C. de C. Lima. Optimization of ANN Applied to Non-linear System Identification Based in UWB. In *Proceedings of the Symposium on Trends in Communications*, IEEE Press, Slovakia, pp. 56–59, 2006.

[7] C. T. Lin, C. S. G. Lee. *Neural Fuzzy Systems: A Neuro-fuzzy Synergism to Intelligent Systems*, Prentice-Hall, Inc., New Jersey, USA, 1996.

[8] G. E. P. Box, G. M. Jenkins. *Time Series Analysis, Forecasting and Control*, Holden Day, San Francisco, USA, 1970.

Bidyadhar Subudhi received the Bachelor degree in electrical engineering from Regional Engineering College Rourkela (presently National Institute of Technology Rourkela), India, Master of technology in control & instrumentation from Indian Institute of Technology, India, in 1994, and the Ph. D. degree in control system engineering from University of Sheffield, UK, in 2003. He worked as a post doctoral research fellow in the Department of Electrical & Computer Engineering, National University of Singapore (NUS), Singapore, in 2005. Currently, he is a professor in the Department of Electrical Engineering in the National Institute of Technology, India. He is a fellow of the Institution of Engineers (India), life member of Systems Society of India and senior member of IEEE. He is serving as a technical committee member, IEEE Intelligent Control Society.

His research interests include system identification, intelligent control, control of mobile and flexible robot manipulators, and estimation of signals & systems.

Debashisha Jena received the Bachelor of electrical engineering degree from University College of Engineering, India, in 1996 and Master of technology in electrical engineering in 2004. He is currently a Ph. D. candidate in the Department of Electrical Engineering, National Institute of Technology, India. He worked as a faculty member in the National Institute of Science & Technology, India, during 2004–2007. He has been awarded a GSEP fellowship in 2008 from Canada for research in control and automation.

His research interests include evolutionary computation and system identification with application to non-linear systems.