

A DPSO-Based NN-PID Controller for MIMO Systems

Tarun Varshney, Ruchi Varshney and Nitin Singh

Abstract The neural networks are generally trained using the standard back propagation (BP) algorithm and its variants. In the BP algorithm, the initial weights are generated randomly which affects the convergence of algorithm, and hence, the algorithm is prone to the problem of local optima. In the proposed work, dynamic particle swarm optimization (DPSO) has been used to initialize the weights of the NN-PID controller for multiple input multiple output (MIMO) systems. The results obtained using the proposed DPSO-based NN-PID controller were compared with the other existing NN-PID control techniques. Simulation results show that the performance of the BP algorithm was significantly improved with the use of DPSO algorithm for initializing the weights.

Keywords Particle swarm optimization • Dynamic particle swarm optimization • Neural network • Back propagation • PID controller

1 Introduction

The simplicity of the PID controllers makes it an excellent choice for applications in the control systems, but the tuning of its parameters is still a cumbersome process. The tuning of parameters also does not guarantee satisfactory performance due to process changes and aging effect. This requires the controller parameters to be

T. Varshney (✉)

Department of Electrical Engineering, Surendera Group of Institutions,
Sri Ganganagar, India
e-mail: t_varshney@yahoo.com

R. Varshney

Department of Electronics & Communication, MIT, Moradabad, India
e-mail: ruchi25varshney@gmail.com

N. Singh

Department of Electrical Engineering, MNNIT, Allahabad, India
e-mail: nitins@mnnit.ac.in

tuned regularly to get the satisfactory performance from the controller, which is a time consuming and impractical approach. Most of the real industrial systems are nonlinear and multi-input and multi-output in nature. These nonlinearities and the interactive behavior among the variables make the limited use of conventional PID controller. The neural networks (NNs) are the powerful tools in controlling such systems due to their universal function approximation capabilities, learning, and generalization properties [1]. The NNs can be used in one of the two ways, either the parameters of PID controller can be adjusted using them [2, 3] or it can act as intelligent controller used for modifying the parameters by using some rules, similar as the idea of an control engineer.

There exist various articles in the literature that used BP algorithm for updating the weights of the NNs, but due to the inherent limitation of the algorithm it suffers from the problem of slow convergence and local optima [4, 5]. Researchers have suggested possible ways to make it faster to overcome the problem of slow convergence [6, 7]. A momentum term has been included in the weight updating method to accelerate the learning procedure. The initial weights in the standard BP and its variants are generated randomly. The sensitivity of these algorithms for the initial weight is very high; therefore, initial weights choice could affect the convergence rate of the algorithm.

Particle swarm optimization (PSO) is similar to other metaheuristics-based algorithms, e.g., genetic algorithm, differential evolution algorithm. In general, the computation power required by these algorithms is huge which motivates the researchers to develop the efficient optimization techniques. PSO is also an evolutionary algorithm which is based on social behavior of animals such as bird flocking, fish schooling, and swarm theory. In this algorithm, in every iterations, the velocity updating of the particle is dependent on three factors, i.e., previous velocity, cognition component, and social component. The idea of PSO is easier to understand, and it has been used for finding the optimal solution to various complex problems [8, 9]. It has motivated the researchers to use it as a tool for optimizing the parameters of NN [10–12].

DPSO is a variant of PSO which is proposed by Nitin Saxena [13], it meets the two prime objectives of the population-based algorithm, i.e., the speed of the convergence and the mechanism for avoiding the local minima, which is difficult to achieve because of contradiction in both the objectives. The basic PSO [14] updates the velocity of the particles by using the global best which also influences the position of the particles and leads to faster convergence, due to which it becomes vulnerable to the problem of local optima especially in the case of multimodal problems [15, 16]. The DPSO variant overcomes the problem of stagnation and local optima and at the same time maintains the fast convergence rate.

This paper presents the use of DPSO algorithm for optimizing the free parameters of NN, i.e., weights. The use of evolutionary programming technique along with NN overcomes the issue of convergence in the NN-PID controller for MIMO systems. Rest of the paper is organized as follows: Sect. 2 describes the structure of NN-PID controller, and further, the input–output relationship of P, I, and D neurons has been defined, Sect. 3 briefs about the concept of classical back propagation algorithm and

BP algorithm with momentum constant, Sect. 4 describes the PSO algorithm and its shortcomings, Sects. 5 and 6 describe the DPSO algorithm and procedure of weight optimization using DPSO algorithm, Sect. 7 shows the simulation results, and finally, paper is finished by a conclusions are drawn in last section.

2 NN-Based MIMO PID Controller

The structure of the NN-based PID controller for the MIMO systems is shown in the Fig. 1 [4]. For n th order system, the whole network is divided into n subnet where each subnet has one input, hidden and output layer. The input layer has two neurons, and the hidden layer has three neurons that mimic the behavior of *Proportional, Integral, and Derivative* actions and considered as *P-neuron, I-neuron, and D-neuron*, respectively, and finally, the output layer has one neuron. The input–output characteristics of the *P-neuron, I-neuron, and D-neuron* are defined as follows:

A. P-neuron

The input for j th *P-neuron* at instant k is as follows:

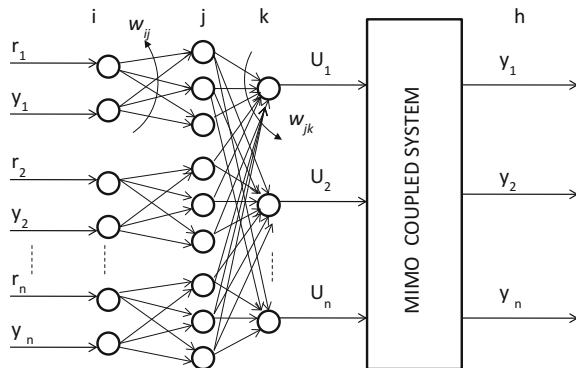
$$u_{pj}(k) = \sum_{i=1}^N w_{ij} \cdot x_i(k) \tag{1}$$

where $i (1, 2), j (= 1, 2, 3), x_i(k)$ are the outputs of N -connected input nodes in input layer and w_{ij} is the weight matrix between input and hidden layer. The relationship between input and output of *P-neuron* is given as (2)

$$x_{pj}(k) = u_{pj}(k) \tag{2}$$

where $x_{pj}(k)$ is the output of *P-neuron*.

Fig. 1 NN-PID controller for MIMO-coupled systems



B. I-neuron

Similarly, relationship of input and output of *I*-neuron is defined as

$$x_{Ij}(k+1) = x_{Ij}(k) + u_{Ij}(k) \quad (3)$$

where $x_{Ij}(k)$ is the *I*-neuron output.

C. D-Neuron

With compliance to derivative action, the characteristic of *D*-neuron is defined as

$$x_{Dj}(k) = u_{Dj}(k) - u_{Dj}(k-1) \quad (4)$$

where $x_{Dj}(k)$ is the *D*-neuron output.

Consequence the output of subnet *Un* is as follows:

$$U_n = w_{jk}^n \cdot X \quad (5)$$

where $X = [x_p^1 x_1^1 x_D^1 \dots x_p^n x_1^n x_D^n]$ and w_{jk}^n is the network weights between hidden and output layer for *n*th subnet.

3 Comparative Study

In order to validate the results obtained from the proposed DPSO-based NN-PID controller for MIMO systems, the following algorithms have been considered for comparison.

A. BP algorithm

The main objective of BP learning algorithm in the NN-PID controller for MIMO systems is to minimize the function given by (6)

$$J = \sum_{h=1}^n E_h = \frac{1}{n} \sum_{h=1}^n \sum_{k=1}^n [i_h(k) - o_h(k)]^2 \quad (6)$$

where $i_h(k)$ and $o_h(k)$ are input and output of the system, respectively. Standard BP algorithm is used for modifying the weights of this controller. After the forward pass, the weights between hidden layer and output layer are modified by:

$$w_{sjh}(n+1) = w_{sjh}(n) - \eta \frac{\partial J}{\partial w_{sjh}} \quad (7)$$

where

$$\frac{\partial J}{\partial w_{sjh}} = -\frac{2}{n} \sum_{k=1}^n [i_h(k) - o_h(k)] \frac{o_h(k+1) - o_h(k)}{u_h(k) - u_h(k-1)} x_{sj}(k) = -\frac{2}{n} \sum_{k=1}^n \partial_h(k) x_{sj}(k) \quad (8)$$

where η represents the learning rate, the serial number of output variables is given by h ($= 1, 2 \dots n$), serial number of hidden neurons in every subnet is given by j ($= 1, 2, 3$), the serial number of the subnet is given by s ($= 1, 2 \dots n$), the training step instant is k , and output of hidden layer neuron is given as x_{sj} . The weights updating law between the input and hidden layer is given by:

$$w_{sij}(n+1) = w_{sij}(n) - \eta \frac{\partial J}{\partial w_{sij}} \quad (9)$$

where

$$\frac{\partial J}{\partial w_{sij}} = -\frac{1}{n} \sum_{h=1}^n \sum_{k=1}^n \partial_h(k) w_{shj} \frac{u_{sj}(k+1) - u_{sj}(k)}{s_{sj}(k) - s_{sj}(k-1)} x_{si}(k) = -\frac{1}{n} \sum_{k=1}^n \partial_{sj}(k) x_{si}(k)$$

where the serial number of input neurons in every subnet is i ($= 1, 2$) and x_{si} is the output of input layer's neurons.

B. BP algorithm with momentum term

The standard BP algorithm despite its popularity has the major drawback of slow convergence rate. To improve the convergence rate, several modifications several modifications have been suggested by researchers. One of the modifications is the addition of the momentum term into standard BP algorithm. The modified weight updating rules [17] are as follows

$$w_{sjh}(n+1) = w_{sjh}(n) - \eta \frac{\partial J}{\partial w_{sij}} + \alpha (w_{sjh}(n) - w_{sjh}(n-1)) \quad (10)$$

$$w_{sij}(n+1) = w_{sij}(n) - \eta \frac{\partial J}{\partial w_{sij}} + \alpha (w_{sij}(n) - w_{sij}(n-1)) \quad (11)$$

where α is the momentum constant which lies between 0 and 1 which represents a rational changes in previous weight to current weight into the current weight change. The addition of this term helps to smoothen the path of decent by controlling the extreme change in the gradient.

4 PSO Algorithm

The objective of any optimization algorithm is to calculate or determine the optimal (i.e., best suited) solution for the given problem under a given set of constraints. Kennedy and Eberhart in the mid-1990s proposed an algorithm which does not require any gradient information and rather works by emulating the collective behavior of bird flocking, particles, and socio-cognition [18] which was named as PSO [18, 19].

PSO is a multi-agent parallel search technique, in which particles are considered as conceptual entities which fly through the multi-dimensional search space. Individual particle in the swarm can be defined on the basis of its position and velocity.

The initial values for $pxbest_j$ and $gxbest$ are taken as $pxbest_j(0)$ and $gxbest(0)$ for all the particles. The velocity and new position of j th particle in the swarm can be calculated using (12) and (13) after each particle knows the best individual particle position in the swarm.

$$v_j^{k+1} = w + v_j^k + c_1 \tau_1 (pxbest_j - x_j^k) + c_2 \tau_2 (gxbest - x_j^k) \quad (12)$$

$$x_j^{k+1} = x_j^k + v_j^{k+1} \quad (13)$$

where v_j^k and x_j^k are the velocity and position of the j th particle at time k , respectively. The individual best position of the j th particle and the global best position of the j th particle in the swarm is given as $pxbest_j$ and $gxbest$, respectively. τ_1 and τ_2 are uniformly distributed numbers within interval [0 1] that determines the impact of $pxbest_j$ and $gxbest$ on the velocity update formula. c_1 and c_2 are constant terms, namely “self-confidence” and “swarm-confidence”, respectively. Initial values of c_1 and c_2 are considered as $c_1 = c_2 = 2$. w is the inertial weight and can be determined using (14).

$$w = w_{\max} - (w_{\max} - w_{\min})Num/Num_{\max} \quad (14)$$

where w_{\min} and w_{\max} are the minimum and maximum values of the w , Num_{\max} is the maximum iteration time of the algorithm, and Num is the current iteration of the algorithm.

5 DPSO Algorithm

The new variant of PSO was proposed by Nitin Saxena [13] which overcomes the two prime objectives of the population-based algorithm, i.e., the speed of the convergence and the method for avoiding the local minima, which is difficult to achieve because of contradiction in both the objectives. The basic PSO [14] updates

the velocity of the particles by using the global best (gbest) which also influences the position of the particles and leads to faster convergence, due to which it becomes vulnerable to the problem of local optima specially in the case of multi-modal problems [15, 16]. Researchers have come up with solution to this problem and proposed various variants of PSO which enhances the performance of algorithm by balancing the parameters [13, 20].

The DPSO variant overcomes the problem of stagnation and local optima and at the same time maintains the fast convergence rate. DPSO does it by keeping the track of the change in the personal best positions of the particles, i.e., pbest, the algorithm tracks time for which the position pbest is not updated and at the same time it also keeps a track of the iteration for which the gbest had not changed. If the pbest and gbest do not change for a predefined number of iterations, then they are replaced by the best positions that they have attained previously.

The pbest and gbest are again monitored to see whether they are able to improve the targets or not if they are able to improve the targets then the replaced values are made permanent, otherwise the original previous values were restored. The repetition of the process is done until the value reaches an optimum point or termination condition is reached. Table 1 shows the different parameters of the DPSO algorithm.

Table 1 Parameters used in DPSO algorithm

Symbol	Description
m	Size of the swarm (no. of particles)
\mathbf{X}_i	Vector representing position of the i th particle, $i = 1$ to m
\mathbf{V}_i	Vector representing velocity of the i th particle, $i = 1$ to m
pBest _{i}	Best position obtained by i th particle in the swarm
gBest	Global best position obtained by the swarm
pBest_count _{i}	Count of iterations for which pBest _{i} have not improved in consecutive generations for particle i
pBest_iter_thresh	Max count of iteration that i th particle waits for improvement in pBest _{i}
pBest_temp _{i}	Temporarily altered vector of pBest _{i}
pBest_chance_count _{i}	Count of iterations for which pBest_temp _{i} do not improve in following generations
pBest_chance_max	Max count of repetitions that pBest_temp _{i} will get to improve pBest _{i}
gBest_h	Vector which stores historical global best values of gBest
gBest_h_maxcount	Max number of features permissible in gBest_h
gBest_count	Count of iterations for which gBest does not improves in consecutive generation for particle i
gBest_iter_thresh	Count of iterations group wait for improvement in gBest (constant)
gBest_temp	Provisionally updated vector of pBest _{i} for particle $i = 1$ to m
gBest_chance_count	Max count of iterations for which gBest_temp does not progress in gBest_chance_max consecutive iterations
gBest_chance_max	Max count of repetitions gBest_temp will get chance to improve gBest

6 Procedure of Weight Initialization of NN-PID Controller Using DPSO Algorithm

The procedure of weight optimization using DPSO is as follows:

1. Initialize randomly the location and speed of every element within the n -dimensionally exploration space. The counter for $pBest_count_i$ and $pBest_chance_count_i$ is set to 0 and -1 , respectively. Similarly, the $gBest_count$ and $gBest_chance_count$ are initialized to 0 and -1 for counting the successive generations for which $gBest$ and $gBest_temp$ have not improved.
2. The fitness of each particle is calculated, and the initial location vector is assigned as $pBest_i$ and location creating best fitness value as $gBest$.
3. The speed of each particle is updated using the (12), and thereafter, the position is updated using (13).
4. The most potential regions identified upto $gBest_h_maxcount$ are identified in the form of $gBest$ and preserved as historic overall best as $gBest_h$.
5. The suitability of each element is computed again with the restructured location; if it improves the personal best of particle, then $pBest_i$ get updated to the X_i , and likewise, if the global best location of present reiteration is superior to $gBest$, then it gets restructured.
6. If the $pBest_i$ are not improved from the previous generation, then the count $pBest_count_i$ is incremented by 1 or else reset to 0. Likewise, if the $gBest$ does not improve from the previous generation, then the count $gBest_count$ is incremented by 1.
7. If for any particle the counts $pBest_count_i$ or $gBest_count$ reaches the threshold values, i.e., $pBest_iter_thresh$ and $gBest_iter_thresh$ correspondingly, then go to next step otherwise check for cessation condition.
8. If the elements whose $pBest_count_i$ and $gBest_count$ have reached the threshold, then their $pBest_i$ and $gBest$ are restructured as $pBest_temp_i$ and $gBest_temp$, respectively.
9. If the elements whose personal best, i.e., $pBest$, is restructured, their $pBest_i$ are changed provisionally to $pBest_temp_i$ for $pBest_chance_max$ number of iterations and checked for improvement as compared to the value before replacement, if it shows improvement, the value is made permanent otherwise it is restored with the original value before exchange. Similarly, process is done for the $gBest$, it is changed with the $gBest_temp$ for $gBest_chance_max$ iterations.
10. The algorithm is terminated if the total number of iterations exceeds the maximum allowed iterations otherwise the process is repeated from step 3.

7 Simulation, Results, and Discussion

In order to test the proposed approach, two case studies of discrete time nonlinear interactive MIMO systems have been presented. All simulations have been performed on MATLAB platform with Intel Core 2 duo processor running at 1.86 GHz and 1 GB of RAM.

Case Study I: A 2×2 discrete time nonlinear-coupled MIMO system [12] is modeled as shown in (15) and (16). A negative unity feedback control topology has been tested to track the set point changes.

$$y_1(k) = \frac{1}{(1 + y_1(k+1))^2} (0.8y_1(k-1) + u_1(k-2) + 0.2u_2(k-3)) \quad (15)$$

$$y_2(k) = \frac{1}{(1 + y_2(k-1))^2} (0.9y_2(k-1) + 0.3u_1(k-3) + 0.2u_2(k-2)) \quad (16)$$

The main objective is to design a NN-PID controller for the discrete time, nonlinear, coupled MIMO system with faster convergence rate for the set point changes.

The parameters of DPSO are initialized as follows:

- The population size is taken as 30
- Fix inertia weight (w) is taken as 0.783
- The acceleration coefficients c_1 and c_2 are taken as $c_1 = c_2 = 1.367$
- Maximum number of iterations is taken as 100
- Maximum number of gBest stored as history in $gBest_h = 10*m$, $pBest_iter_thresh = gBest_iter_thresh = 5*m$ (dimension).

A control topology with negative feedback has been implemented using NN-PID controller where objective is to track the set point of 0.7 and 0.4 for input-1 and input-2, respectively, with respect to time (seconds). Initially, weights have been generated randomly between 0 and 1 and learning rate and momentum constant are chosen, 0.40 and 0.04, respectively. Parameters of PID controller $Kp_1 = 0.10$, $Ki_1 = 0.05$, $Kd_1 = 0.001$ and $Kp_2 = 0.10$, $Ki_2 = 0.05$, $Kd_2 = 0.001$ are chosen.

The training of the NN-PID controller is done using four different techniques. Firstly, standard BP algorithm is used for initializing the network weights randomly. Secondly, standard BP algorithm with random initialization is used but the outputs of P-neuron, I-neuron and D-neuron have been multiplied by constant factors of 1, 10, and 1.5, respectively. Thirdly, standard BP algorithm with momentum constant with random initialization has been used, and finally, standard BP algorithm with DPSO-based initialization has been chosen and tested.

Figures 2 and 3 show the step responses of the system for the input-1 and input-2, and corresponding controlled inputs have been shown in Figs. 4 and 5. It is evident from these figures that with the controlled inputs the proposed scheme tracks the set point changes faster than the other existing techniques. Significant

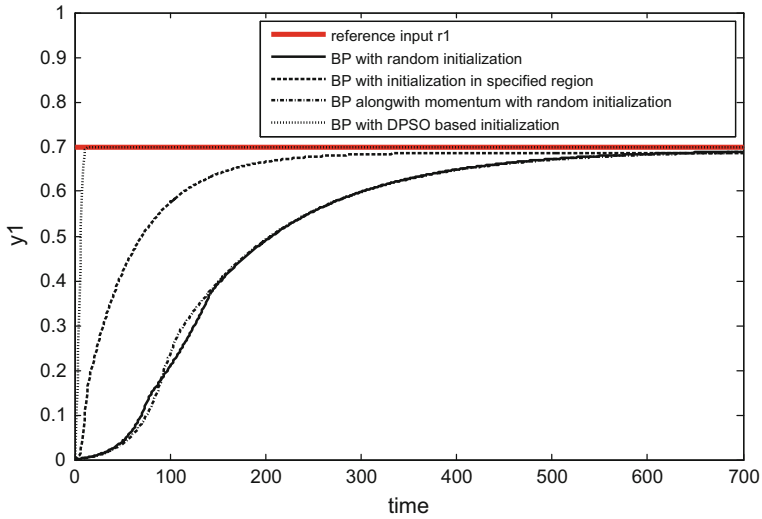


Fig. 2 System response for set point change in input 1 (case study I)

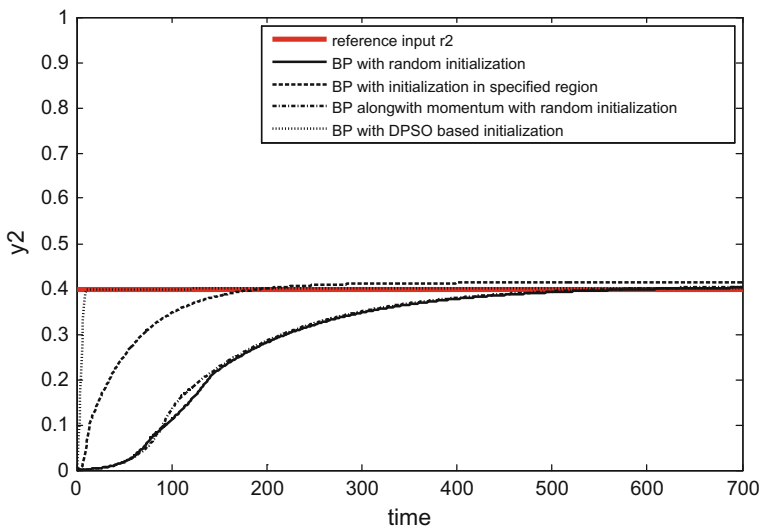


Fig. 3 System response for set point change in input 2 (case study I)

improvement is also seen in the square error convergence with the use of DPSO algorithm for weight initialization which is shown in Fig. 6. The improvement in rise time and settling time is also very clearly visible in the simulation results.

Case Study II: A 3×3 discrete time nonlinear-coupled MIMO system whose input-output relation is given by (17)–(19)

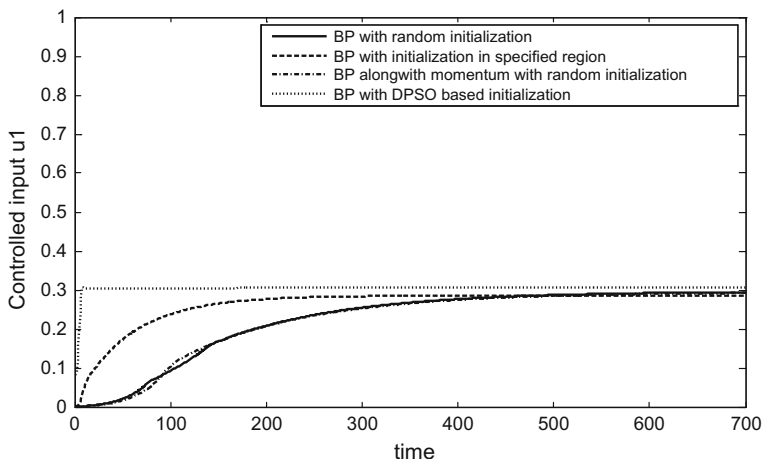


Fig. 4 Controlled input 1 (case study I)

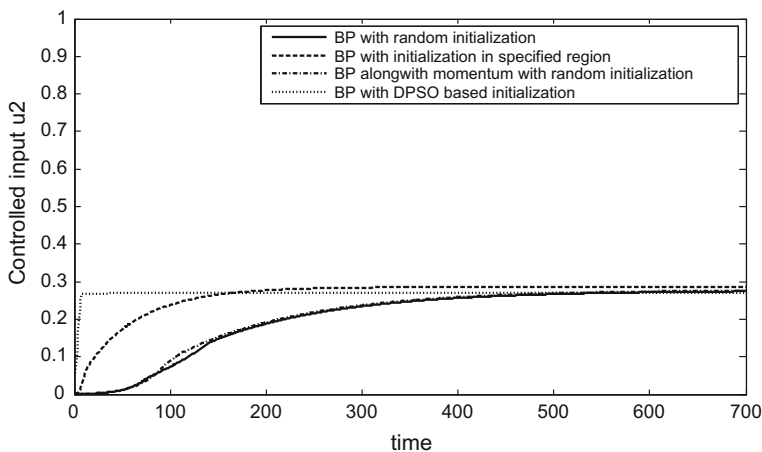


Fig. 5 Controlled input 2 (case study I)

$$y_1(k) = 0.4y_1(k-1) + \frac{u_1(k-1)}{1 + u_1^2(k-1)} + 0.2u_1^3(k-1) + 0.5u_2(k-1) + 0.3y_2(k-1) \tag{17}$$

$$y_2(k) = 0.2y_2(k-1) + \frac{u_2(k-1)}{1 + u_2^2(k-1)} + 0.4u_2^3(k-1) + 0.2u_1(k-1) + 0.3y_3(k-1) \tag{18}$$

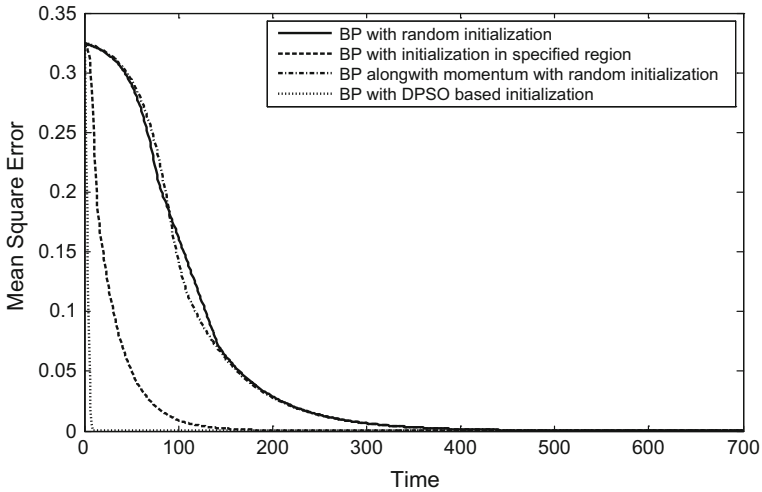


Fig. 6 Convergence of mean square error (case study I)

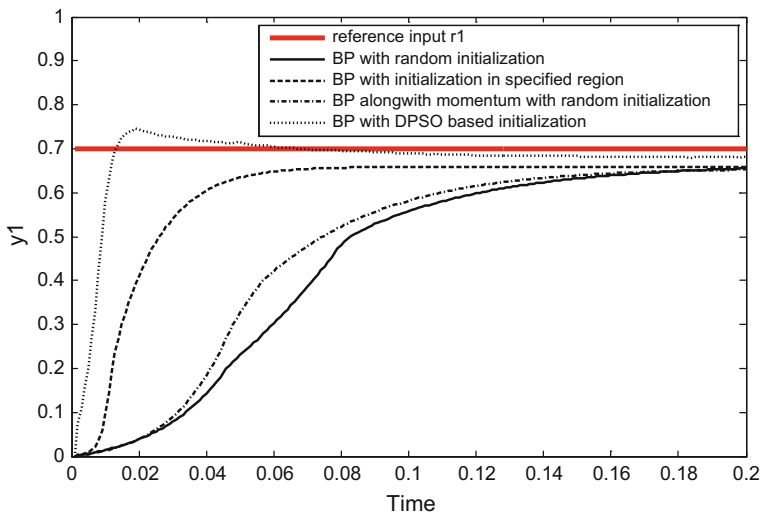


Fig. 7 System responses for input 1 (case study II)

$$y_3(k) = 0.2y_3(k-1) + \frac{u_3(k-1)}{1 + u_3^2(k-1)} + 0.4u_3^3(k-1) + 0.4u_2(k-1) + 0.3y_1(k-1) \tag{19}$$

The objective here is same as the earlier case study. Set points of 0.2, 0.4, and 0.7 are used for input-1, input-2, and input-3, respectively. Initial weights have been

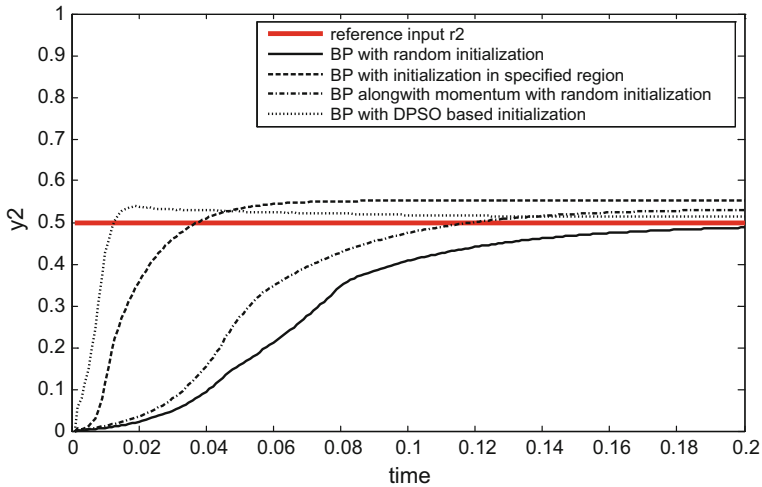


Fig. 8 System responses for input 2 (case study II)

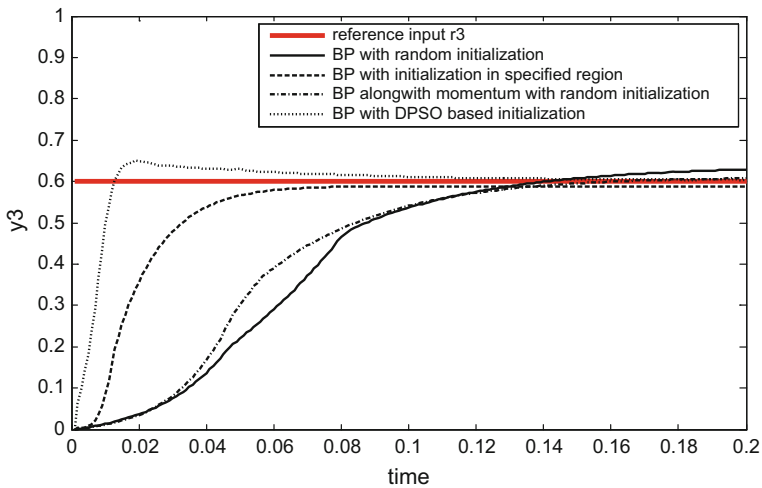


Fig. 9 System response for input 3 (case study II)

generated randomly between 0 and 1, and learning rate and momentum constant are same as case study I. Controller parameters ($Kp_1 = 0.3, Ki_1 = 0.15, Kd_1 = 0.1, Kp_2 = 0.3, Ki_2 = 0.15, Kd_2 = 0.1, Kp_3 = 0.3, Ki_3 = 0.15, Kd_3 = 0.1$) have been chosen, and ten hidden neurons have been determined by observing the performance measure.

The parameters of DPSO algorithm that are adopted earlier remain unchanged. The responses of the system for step change to all the inputs are shown in Figs. 7, 8, and 9.

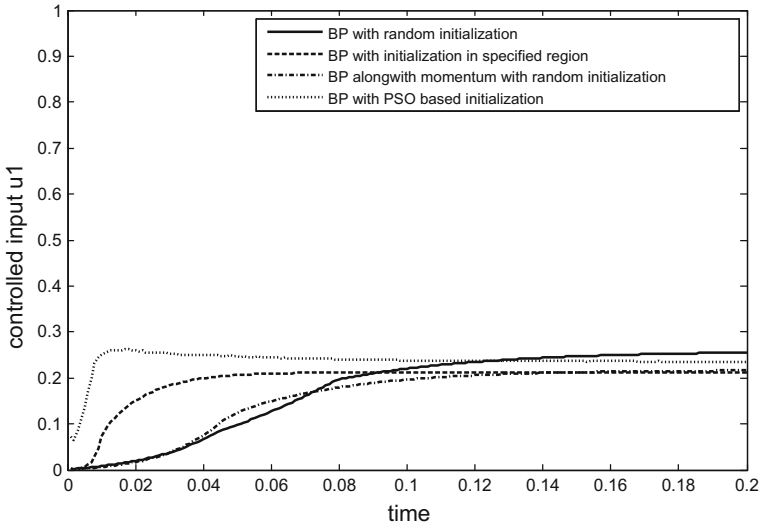


Fig. 10 Controlled input 1 (case study II)

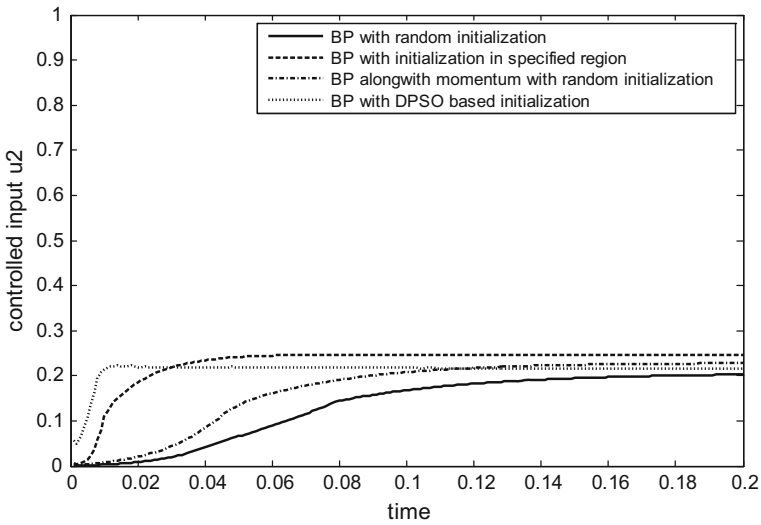


Fig. 11 Controlled input 2 (case study II)

The step response of the system to the controlled inputs is shown in Figs. 10, 11 and 12. It is observed from the results that with feasible-controlled inputs, the NN-based PID controller trained with BP algorithm using DPSO-based weights initialization is capable of tracking the set point changes faster with least steady error. The rise time and settling time have also been significantly improved (Fig. 13).

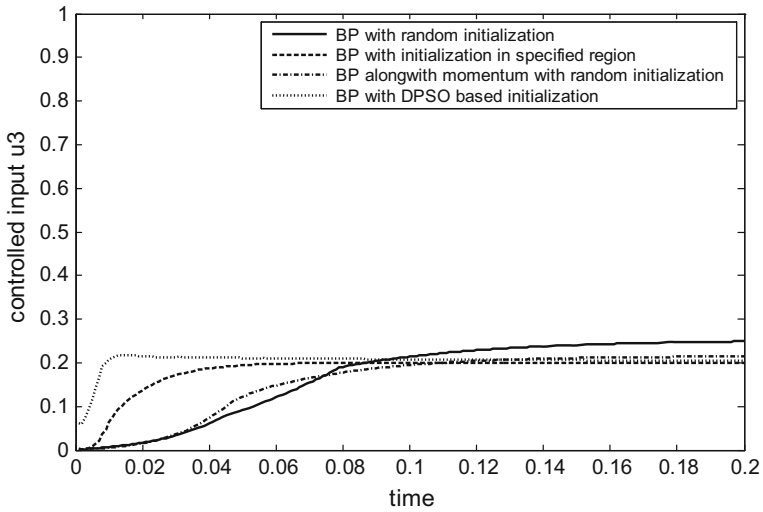


Fig. 12 Controlled input 3 (case study II)

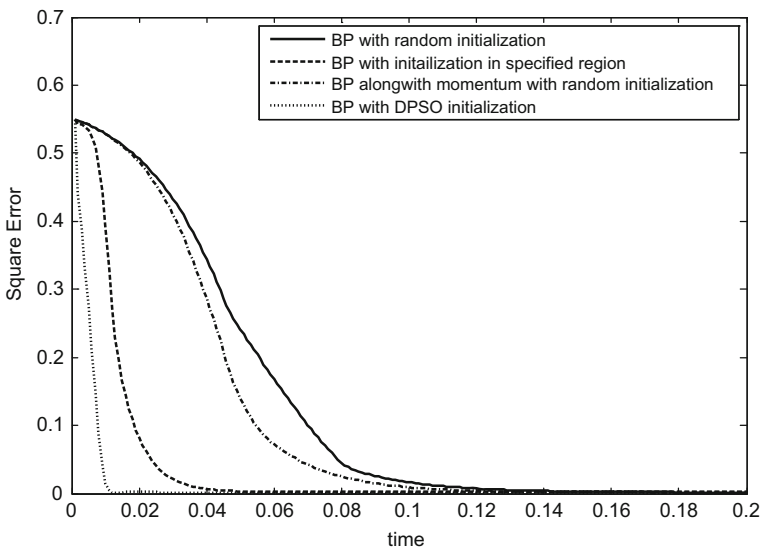


Fig. 13 Convergence of square errors (case study II)

In the case study II, it is clearly observed that the transient response and rate of convergence have been improved significantly with feasible-controlled inputs.

8 Conclusion

The structure of NN-based PID controller using *P-neuron*, *I-neuron*, and *D-neuron* has been explored for controlling the nonlinear-coupled MIMO systems. The training of the NN-based controller is done by using standard BP algorithm, and with momentum term added to BP algorithm. The initial weights of the network were chosen randomly. The convergence of the BP algorithm is further improved by utilizing the DPSO algorithm for optimizing the weights of the NN-based PID controller. The case studies of two discrete nonlinear-coupled MIMO systems are presented, and the simulated results are fairly compared with the results of the other existing techniques. On the basis of results, it is concluded that proposed approach not only provides better control of the nonlinear-coupled MIMO systems but also satisfies the minimum mean square error criteria with lesser settling and rise time. The convergence rate of the proposed algorithm is found to be better than both standard BP and BP with momentum term.

References

1. T. Varshney and S. Sheel, "Approximation of 2D function using simplest neural networks: A comparative study and development of GUI system," IEEE International conference on Power, control and embedded system, MNNIT Allahabad, INDIA, 2010, pp. 1–4.
2. T. Varshney and S. Sheel, "A new online tuning approach for PID control of multivariable systems using diagonal recurrent neural Network," IEEE International Conference on Control System, Computing and Engineering, (25–27 November 2011), Pinang, Malaysia, pp 317–320.
3. T. Varshney and S. Sheel, "A Morlet wavelet neural network-based online identification and control of coupled MIMO systems," *International Journal of Automation and Control*, vol. 6, no. 3/4, p. 246, 2012.
4. Huailin Shu, Xiucui Guo, and Hua Shu, "PID neural networks in multivariable systems," IEEE International Symposium on Intelligent Control, 2002, pp. 440–444.
5. N. Singh, D. K. Chaturvedi, and R. K. Singh, "A Modified Error Function GNN For Load Frequency Control of Multi-area Power System," in *Proceedings of the 2010 International Conference on Artificial Intelligence, ICAI 2010, July 12–15, 2010, Las Vegas Nevada, USA, 2 Volumes*, 2010, pp. 353–359.
6. L. Behera, S. Kumar, and A. Patnaik, "On Adaptive Learning Rate That Guarantees Convergence in Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1116–1125, Sep. 2006.
7. S. Sheel, T. Varshney, and R. Varshney, "Accelerated learning in MLP using adaptive learning rate with momentum coefficient," IEEE International Conference on Industrial and Informatics system, Peradeniya, Sri Lanka (August 8–11, 2007), pp 307–310.
8. Eberhart and Yuhui Shi, "Particle swarm optimization: developments, applications and resources," 2001, vol. 1, pp. 81–86.
9. R. A. Krohling and L. dos Santos Coelho, "Coevolutionary Particle Swarm Optimization Using Gaussian Distribution for Solving Constrained Optimization Problems," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 6, pp. 1407–1416, Dec. 2006.

10. J. L. Cao, J. M. Yin, J. S. Shin, and H. H. Lee, "BP network modified by particle swarm optimization and its application to online-tuning PID parameters in idle-speed engine control system," in *2009 ICCAS-SICE*, 2009, pp. 3663–3666.
11. J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1026–1037, Feb. 2007.
12. C. Jin, S.-W. Jin, and L.-N. Qin, "Attribute selection method based on a hybrid BPNN and PSO algorithms," *Applied Soft Computing*, vol. 12, no. 8, pp. 2147–2155, Aug. 2012.
13. N. Saxena, A. Tripathi, K. K. Mishra, and A. K. Misra, "Dynamic-PSO: An improved particle swarm optimizer," *IEEE Congress on Evolutionary Computation*, Sendai, Japan, 2015, pp. 212–219.
14. R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, 1995, vol. 1, pp. 39–43.
15. Jang-Ho Seo, Chang-Hwan Im, Chang-Geun Heo, Jae-Kwang Kim, Hyun-Kyo Jung, and Cheol-Gyun Lee, "Multimodal function optimization based on particle swarm optimization," *IEEE Transactions on Magnetics*, vol. 42, no. 4, pp. 1095–1098, Apr. 2006.
16. J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
17. M. Negnevitsky and M. Ringrose, "Accelerated learning in multi-layer neural networks," 1999, vol. 3, pp. 1167–1171.
18. J. Kennedy, J. F. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm intelligence*. Morgan Kaufmann, 2001.
19. T. Zeugmann *et al.*, "Particle Swarm Optimization," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Boston, MA: Springer US, 2011, pp. 760–766.
20. Y. Shi and R. Eberhart, "A modified particle swarm optimizer," 1998, pp. 69–73.