5th SASTech
Iran | Mashhad
May 12 - 17 | 2011
5th Symposium on Advances in Science & Technology

An improved PID neural network controller for long time delay systems using particle swarm optimization algorithm

A. Lari[1], A. Khosravi[1] and A. Alfi[2]

[1] Faculty of Electrical and Computer Engineering, Noushirvani University of Technology, Babol 47135-484, Iran
[2] Faculty of Electrical and Robotic Engineering, Shahrood University of Technology, Shahrood 36199-95161, Iran.

Paper Reference Number: 0101-491
Name of the Presenter: A. Lari

Abstract

Conventional PID controller is the most popular controller in various field of industry. In spite of strong ability, this controller cannot be usually used for long time delay systems. There is a kind of neural networks called PID neural network (PIDNN) that utilizes the advantages of both PID controller and neural network simultaneously. PIDNN's weights were in first adjusted by back propagation (BP) algorithm. BP algorithm ensures the final convergence, but the critical drawback is that the study training costs a long time. The convergence into local extremum is also possible. Hence, another method must be opted to optimize the network's weight. This paper proposes a novel PIDNN without saturate surface namely PIDNN-PSO which its weights are adjusted using particle swarm optimization (PSO). PSO algorithm is an evolutionary optimization algorithm that due to the ease of implementation and fast convergence speed has been widely applied in many areas. The proposed controller is utilized as a controller for long time delay systems. The performance of the proposed controller is compared with the controller is designed by PIDNN-BP algorithm. Simulation results demonstrate the effectiveness of the proposed algorithm.

Key words: PID controller, PSO, BP algorithm, PIDNN

1. Introduction

Most of industry processes include a long time delay system. Therefore a controller design for a long time delay system has a great deal of importance. Addition long time delay, these systems have generally larger overshoot, longer settling time and are not stable.

Conventional PID controller is the most popular controller in various fields of industry. High efficiency and ease of implementation cause this controller be a powerful controller in industry. In spite of these abilities, they can not be employed for long time delay systems because the parameters of PID controller are difficult to choose. For using the advantages of

this controller in industry processes, Huailin and Youguohas (2000) proposed a new kind of controller namely PIDNN. PIDNN is a combination of PID controller and neural network. The main problem in neural network is the training technique. One of the most popular techniques to training a feed-forward neural network is back propagation algorithm (BP). This algorithm is completely dependent on the initial guess. A proper initial guess near the global extremum lead to a fast convergence otherwise an improper selection specially in problems with a lot of local extremum cause the algorithm trapped in a local extremum.

A particle swarm optimization (PSO) algorithm for training of the feed-forward neural networks has proposed by Song Shao et. al (2007). PSO algorithm is a kind of evolutionary algorithm. One of the main properties of evolutionary algorithm is that initial guess cannot be highly effective on them. Among evolutionary algorithms, PSO algorithm has been become available and promising techniques for real world optimization problems. Due to the simple concept, easy implementation and quick convergence, nowadays PSO has attracted much attention and wide applications in various fields, see Franken and Engelbrecht (2005), Ho et. al (2008), Liu et. al (2007) and Wei-Der and Shun-Peng (2010).

Based on aforementioned before, in this paper, PSO algorithm is employed to train a novel PIDNN without saturate surfaces namely PIDNN-PSO. Saturate surfaces are essential elements for convergence when training of the PIDNN is based on BP algorithm. Hence, these can be removed while a PSO algorithm applies for training. Removing these surfaces causes a faster tracking of reference signals in a control loop .On the other hand, saturate surfaces prevent PIDNN to track larger signals. The proposed controller is used as a controller for long time delay systems.

The rest of this paper is organized as follow: In section 2 structure of PIDNN is illustrated. Disadvantages of using BP algorithm in a PIDNN is discusses in section 3. Section 4 discusses the PSO algorithm. Section 5 is dedicated to training of the proposed controller. Finally, in section 6 simulation result is presented.

2.Structure of PIDNN

A PIDNN is a three layers neural network .These three layers are input, hidden and output layers. Number of neurons in each layer is 2, 3 and 1 respectively. There are three types of neuron in a PIDNN: Proportional neurons, derivative neurons and integral neurons. Input layer consist of two proportional neurons whereas hidden layer consists of one proportional, one derivative and one integral neuron. Finally, output layer consists of one proportional neuron. Figure 1 depicts the general structure of PIDNN as a controller in a closed-loop system. The input-output relationship for a proportional neuron is as follows:

$$x(k)=\begin{cases} 1 & u(k)>1 \\ u(k) & -1\le u(k)\le 1 \\ -1 & u(k)\le -1 \end{cases} \tag{1}$$

where $u(k)$ is the input of neuron and $x(k)$ is the output of neuron.

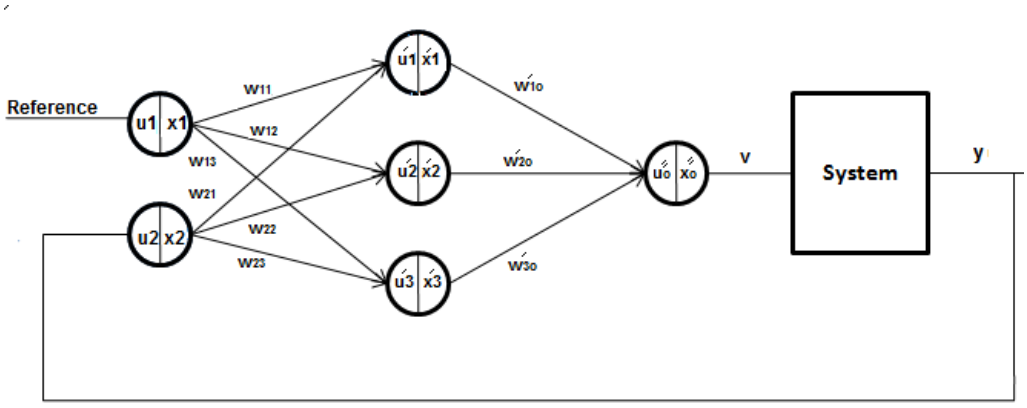

Fig 1: General structure of PIDNN in a closed-loop system

Integral and derivative neurons have the following input-output relationship respectively.

$$x(k)=\begin{cases}1 & x(k)>1\\ x(k-1)+u(k) & -1\le x(k)\le 1\\ -1 & x(k)\le -1\end{cases} \tag{2}$$

$$x(k)=\begin{cases}1 & x(k)>1\\ u(k)-u(k-1) & -1\le x(k)\le 1\\ -1 & x(k)\le -1\end{cases} \tag{3}$$

In this structure, there are 9 unknown weights for a PIDNN which must be determined.

3. Disadvantages of using BP algorithm in a PINNN

Generally BP algorithm has some disadvantages like speed convergence and selection of proper initial guess. But, this algorithm have been confronted with new problems when be used in training of a PIDNN. These disadvantages can be summarized as follows:

A) One of the essential elements in BP algorithm is the calculation of gradient. In formula that is extracted for applying BP in a PIDNN (Huailin and Youguo, 2000), these relationships related to the calculation of gradient is obtained:

$$\frac{\partial y}{\partial v}\approx\frac{\Delta y}{\Delta x_o^{''}}=\frac{y(k+1)-y(k)}{x_o^{''}(k+1)-x_o^{''}(k)} \tag{4}$$

$$\frac{x_j^{'}(k+1)-x_j^{'}(k)}{u_j^{'}(k+1)-u_j^{'}(k)} \tag{5}$$

In equation 4, $x_o^{''}(k)$ is the output of proportional neuron in output layer, v is the output of PIDNN and y is the output of closed-loop system. In equation 5 $x_j^{'}(k)$ and $u_j^{'}(k)$ are outputs and inputs of neurons in the hidden layer. These equations are an approximation of derivation for calculation of gradient. So an error in calculation happens in this stage. On the other hand, in a closed-loop system when states approach to steady state condition, denominator of equations 4 and 5 tend to zero and calculation of these equations is not feasible. Therefore, necessary for an algorithm while gradient is not applied for PSO algorithm.

B) Equations 1-3 include saturate surfaces. Saturate surfaces are essential elements for convergence when training of the PIDNN is based on BP algorithm. But, they have some

disadvantages. For example when the output of the output and hidden layers is saturated, equations 4 and 5 are calculated with a larger error. Meanwhile saturate surfaces causes the output of PIDNN has limit amount and not to be able to track larger reference signals in a closed-loop system. Even for small reference signals, saturate surface causes a slow tracking be occurred. In addition, generally a relatively large overshoot in step response in this situation is inevitable. That's why in this paper a PIDNN without saturate surfaces is proposed.

To overcome these shortages, the input-output relation of neurons is modified as follows:

$$x(k) = u(k) \tag{6}$$

$$x(k) = u(k) - u(k-1) \tag{7}$$

$$x(k) = u(k) - u(k-1) \tag{8}$$

Equations 6-8 demonstrate that the input-output relationship of proportional, derivate and integral neurons in new PIDNN, respectively.

3. PSO Algorithm

PSO is originally attributed to Kennedy and Eberhart (1995) based on the social behavior of collection of animal such as birds flocking and fish schooling. In PSO algorithm, each individual of the swarm called particle, remembers the best solution found by itself and the whole swarm along the search space. The particles move along the search space and exchange information with other particles according to the following equations:

$$V_{id} = wV_{id} + c_1 r_1 (Pid - X_{id}) + c_2 r_2 (P_{gd} - X_{id}) \tag{9}$$

$$X_{id} = X_{id} + V_{id}, \quad for \quad d = 1,2,\ldots,N \;\; i = 1,2,\ldots,S \tag{10}$$

where X_{id} represents the current position of particle, P_{id} is the best remembered individual particle position, P_{gd} denotes the best remembered swarm position, c_1 and c_2 are acceleration coefficients and are known as the cognitive and social parameters, respectively, r_1 and r_2 are random numbers between 0 and 1 and w is inertia weight which is used to balance the global and local search abilities. The inertia weight is critical for the performance of PSO, which balances global exploration and local exploitation abilities of the swarm. A big inertia weight facilitates global search, but it makes the particle long time to converge. Conversely, a small inertia weight makes the particle fast converge, but it sometimes leads to local optimal.

In an empirical study on PSO, Shi and Eberhart (1998) claimed that a linearly decreasing inertia weight can improve local search towards the end of a run, rather than using a constant value throughout. A following decreasing function for the dynamic inertia weight can be devised:

$$w = (iter_{\max} - iter_{cur}) \cdot \left(\frac{w_{initial} - w_{final}}{iter_{\max}} \right) + w_{final} \tag{11}$$

where $w_{initial}$ and w_{final} represent the initial and final inertia weights respectively at the start of a given run, $iter_{\max}$ the maximum number of iterations in a offered run, and $iter_{cur}$ the current iteration number at the present time step.

However, global search ability at the end of the run may be inadequate due to the utilization of a linearly decreasing inertia weight .The PSO may fail to find the required optimal in cases when the problem is too complicated. But to some extent, this can be overcome by employing a self-adapting strategy for adjusting the acceleration coefficients. Suganthan (1999) applied the optimizing method that make the two factors decrease linearly with the increase of iteration numbers, but the results were not as good as the fixed value 2 of c_1 and c_2 . Ratnaweera et. al (2004) improve the convergence of particles to the global optima based on the way that make c_1 decrease and c_2 increase linearly with the increase of iteration numbers. The c_1 and c_2 is given by the following equations:

$$c_1 = c_{1s} + iter_{cur}(c_{1e} - c_{1s}) / iter_{\max} \tag{12}$$

$$c_2 = c_{2s} + iter_{cur}(c_{2e} - c_{2s}) / iter_{\max} \tag{13}$$

where c_{1s} and c_{2s} represent the initial value of c_1 and c_2. Moreover, c_{1e} and c_{2e} represent the final value of c_1 and c_2.

It is noticeable to recall that some of the major differences between PSO and "conventional" optimization algorithms are as follows:

- The objective function's gradient is not required; they only use function values.
- The search is guided probabilistically.

The optimization algorithm operates on a population of solutions (rather than a single solution).

4. Training of the new PIDNN using a pso approach

As mentioned in section 2, BP algorithm can ot be efficient in training of a PIDNN. Therefore, the training of the novel PIDNN (a PIDNN without saturate surfaces) is described. The procedure for this training can be summarized as follows:

Step 1: Initialize the positions and velocities of some particles randomly in the search space. A particle for PIDNN is a 9 dimensional vector as follows:

$$X = [w_{11}\ w_{12}\ w_{13}\ w_{21}\ w_{22}\ w_{23}\ w'_{1o}\ w'_{2o}\ w'_{3o}] \tag{14}$$

where X is the position of a particle in the search space that consists of the PIDNN's weights according to figure 1.

Step 2: Evaluate each initialized particle's fitness value. The fitness function for a closed-loop system can define as the difference between the reference signal and the output of the closed-loop system. Set P_{id} for each particle as its initialized position and P_{gd} as the best initial swarm position.

Step 3: If the maximal iterative generations are arrived, go to step 6, else to step 4.

Step4: Calculate the best remembered individual particle position P_{id} and the best remembered swarm position P_{gd} based on the fitness function. The position and the velocity of all particles are updated according to equations 9 and 10, respectively. Then a group of new particles are generated.

Step 5: Calculate the new amount of w, C_1 and C_2 according to equations 11, 12 and 13 for next iteration, respectively.

Step 6: If stopping criteria is not satisfied, go to step 3. $_{t}$

5. Simulation results

In order to show the performance of proposed controller (PIDNN-PSO), the following long time delay system is considered (Huailin and Youguo, 2000):

$$y(k+1) = .368y(k) + .632v(k-10) \tag{15}$$

It is clearly that this system has a ten seconds delay in output. PIDNN was used by Huailin and Youguo (2000) as a controller for this system which is trained by a BP algorithm. Figure 2 shows a step response obtained by this technique (Huailin and Youguo, 2000)**.** According to figure 2, it can be seen that this step response has a rise time about 20 seconds and a settling time about 48 seconds.

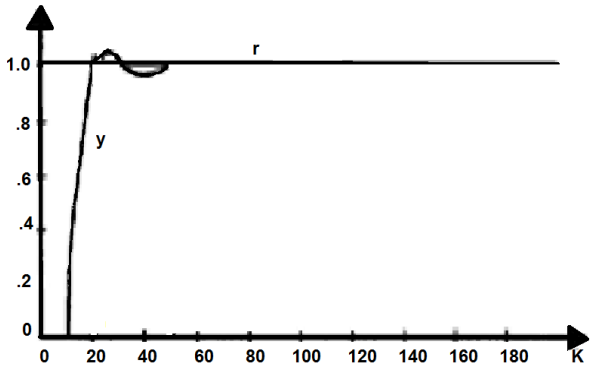

Fig 2: Step response of the closed-loop system with PIDNN controller trained by BP algorithm (Huailin and Youguo, 2000)

Simulation results for step response of this system with PIDNN-PSO controller is demonstrated in figure 3. In this figure, the rise and settling times decrease to a number about 14 and 23 second, respectively. In the same way, simulation result for step response with a magnitude of 10 is demonstrated in figure 4.

It can be seen that unlike the PIDNN with saturate surface the PIDNN-PSO is able to track reference signals with larger amount. The obtained weights for two step responses with PIDNN-PSO controller is listed in table 2.

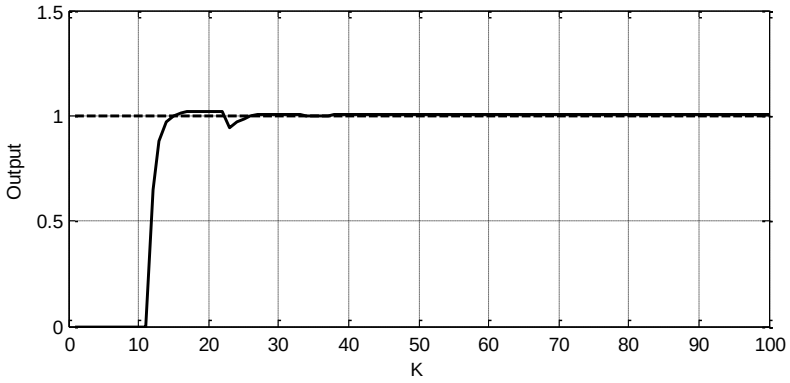

Fig 3: Unit step response with NPIDNN-PSO controller

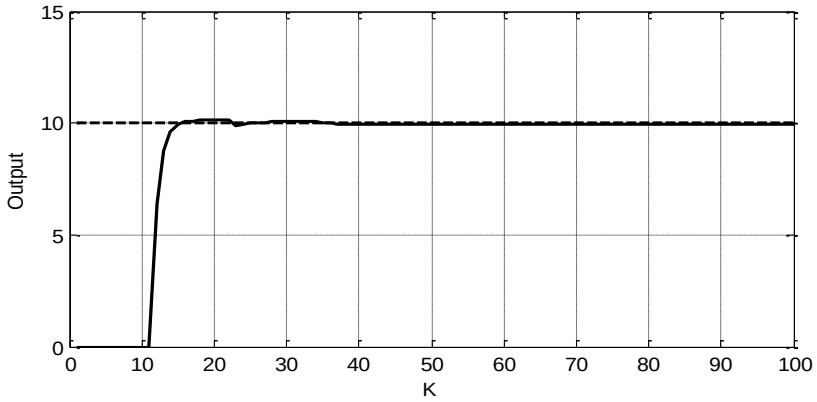

Fig 4: Step response with magnitude of 10 with PIDNN-PSO controller

Table 2: The results of weights obtained by PIDNN-PSO controller

Weights	Step response with magnitude of 1	Step response with magnitude of 10
W_{11}	-0.082308	-0.500000
W_{12}	-0.117576	0.500000
W_{13}	-0.068082	0.500000
W_{21}	-0.118569	-0.500000
W_{22}	0.139848	-0.500000
W_{23}	-0.234682	0.500000
$W_{1o}^{'}$	0.024774	0.500000
$W_{2o}^{'}$	0.001515	0.500000
$W_{3o}^{'}$	0.014739	0.162482

6. Conclusion

This paper proposed a novel PIDNN namely PIDNN-PSO which can be used as a controller in long time delay systems. Unlike BP, PSO algorithm has a randomly search. Moreover, the different initial guess can not highly affect on its convergence. Based on this, in the proposed controller, PSO algorithm was employed to training PIDNN. Simulation results demonstrated the effectiveness of proposed algorithm.

References

Huailin, S., & Youguo, P. (2000). PID neural networks for time-delay systems, Proceedings of the 7th International Symposium on Process Systems Engineering. Keystone, Colorado, USA. pp.859-862.

Song Shao, Z., & Zhang Li, B., & Shu, H. (2007). Application of particle swarm optimization algorithm in training forward neural network, Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.

Franken, N., & Engelbrecht, A. P. (2005). Particle swarm optimization approaches to coevolve strategies for the iterated prisoner's dilemma, *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 562–579.

Ho, S.Y., & Lin, H.S., Liauh, W.H., & Ho, S.J. (2008). OPSO: orthogonal particle swarm optimization and its application to task assignment problems, *IEEE Trans. Syst., Man, Cybern. A,*vol. 38, no. 2, pp. 288–298.

Liu, B., & Wang, L., & Jin, Y. H. (2007). An effective PSO-based mimetic algorithm for flow shop scheduling. *IEEE Trans. Syst., Man, Cybern. B.*, vol. 37, no. 1, pp. 18–27.

Wei-Der, C., & Shun-Peng, S. (2010). PID controller design of nonlinear systems using an improved particle swarm optimization approach.

Kennedy J., & Eberhart R. (1995). Particle swarm optimization, In*: Proc IEEE int conf neural networks*, vol. IV, Perth, Australia; p. 1942–48.

Shi, Y. & Eberhart, R. C. (1998). Parameter selection in particle swarm optimization, *In Proceedings of Evolutionary Programming VII*. pp. 591–600, Springer, New York.

Suganthan, P. N. (1999). Particle swarm optimizer with neighborhood operator, In *Proceedings of IEEE International Conference on Evolutionary Computation. Washinington D.C., IEEE Press*, pp. 1958–1962.

Ratnaweera, A. S., & Halgamuge, K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time varying acceleration coefficients, IEEE Transactions on Evolutionary Computation **8** (3), 240–255.