PSO-RBNN Based Control Design for Trajectory Tracking

NEHA KHURANA
Maharishi Dayanand University, Haryana, INDIA.

Abstract: Inspite of so universally accepted, control performance by NN depends on many of the varying factors such as output weights. To ensure the functional accuracy of the NN, it is required to have an defined value of these performance effecting factors. Control scheme proposed in this paper uses an emerging optimization technique naming, PSO to get the optimal value of the parameters, naming spread factor and weights of output layer in RBNN. Thus, this hybrid controller possesses the advantageous qualities of RBNN and PSO both. For the further improvement in the basic PSO algorithm, inertia weight factor of PSO is made adaptive.This projected controller has been verified by comparing it with a basic PSO and the basic RBNN controller for the trajectory tracking control of a 2-DOF remotely driven robotic manipulator. To check the robustness of the controller its performance has been checked by incorporating uncertainties naming payload masses and friction. Appropriate conclusions have been drawn in last.

Keywords: Radial Bias Neural Network (RBNN), Particle Swarm Optimization (PSO), Evolutionary Neural Network (ENN), Hybrid Intelligent Controller, Remotely Driven Links Manipulator, Motion Control of Non-linear systems

Received: June 27, 2021. Revised: March 21, 2022. Accepted: April 25, 2022. Published: June 3, 2022.

.

1. Introduction

With the increase in present applications of computers and its everyday increasing future prospects; areas of artificial intelligence based controllers have been expanded exponentially. Since the last few decades, because of highly non-linear mapping capabilities, neural network is one of the most widely used AI techniques [1]. There are a wide number of types of neural networks proposed in literature. Each one has its own advantages and disadvantages. In terms of time-taken, accuracy in results and non-linear mapping capabilities for non linear systems like motion control of robotic manipulator, RBNN (Radial Bias Neural Network) is found to superior when compared with back propagation neural network [2-8]. RBNN is given by Broomhead and Lowe [9], and its interpolation and generalization properties are thoroughly investigated in [10, 11]. As stated, although RBNN is one of the commonly used NN based control scheme for the non linear, time varying control system, yet the accuracy in performance of RBNN depends mainly upon the specific values of some of its parameters. A few of the important performance deciding RBNN factors are spread factor, (σ_j) and weights from hidden to output layer, (w_{jk}). Most favorable value of these parameters can be chosen by either some expert's experience or by trial and error (TAE) method. This limitation of RBNN restricts its use to an expert or by using time consuming, tedious and frustrating TAE method by an amateur. This limitation of RBNN restricts its use or deteriorates its performance. From the above discussions it can also be inferred that improvements in RBNN can be made by choosing its accurate parameters. One of the global optimization techniques like PSO can be very constructive to search out the optimized value of RBNN parameters. PSO, developed by Kennedy and Elbert, in 1995 [12] is based on the simulation of simplified animal social behavior such as fish schooling, bird flocking etc.. Stochastic based search algorithm PSO is a global searching technique with simplicity and practicability and has been widely used in recent years to get the optimal solutions [13].

Henceforth, in this paper, to develop the proposed hybrid controller two important techniques naming Particle Swarm Optimization (PSO) and the Neural Network (NN) have been combined. This type of control schemes, taking advantageous features of both the above mentioned PSO and NN intelligent techniques and is named as Evolutionary Neural Networks (ENN). By choosing PSO, auto adaptability quality is developed in the RBNN [14]. In [15-16] such adaptive hybrid controllers have been shown better control performance as compared to other prevailing controllers. ENN has been called as the next generation Neural Networks [17]. Moreover, some improvements in PSO further add on performance quality as, Cao et al. [18] and Shi et al. [19] used modified PSO to optimize RBNN and obtained effective results.

Robotic manipulator is a highly non-linear, time-varying and highly coupled system. For a manipulator, almost all kinds of control techniques naming classical PD, PID, SMC, NN, etc. have been compiled in literature [20, 21 and references there in]. But because of the presence of the various structured and unstructured uncertainties in the model dynamics; still the thrust for a perfect and accurate controller is there.

In this paper, controller used is the hybrid of two model free control techniques naming, PSO and NN. PSO is used to get the finest possible performance deciding RBNN constants, naming spread factor (σ_j) and weights of output layer (w_{jk}). Thus, a successful attempt to make a controller with great control outputs for a manipulator has been made. For further improvement in the control scheme, inertia weight factor of PSO is made adaptive. For simulation purpose, a 2-DOF robotic manipulator having planar elbow with remotely driven links manipulator has been taken here. This type of model is with gear, linear, well understood as the non-linear coupling between the motors has been reduced. On the other hand, this gear introduces friction, compliance, backlash in the dynamics. It has been observed from the literature survey that a very few controllers has been implemented for trajectory tracking control of this planar elbow with remotely driven links manipulator. Performance of the controller with this manipulator has been checked in presence of payload mass changes and the unavoidable friction.

Furthermore, the paper is organized as follows: Section II deals with manipulator dynamics and the fundamentals of the controllers; next, Section III contains the basic scheme of the proposed controller of the paper. Simulation example and results are given in Section IV. Finally, conclusions have been complied in Section V.

2. Fundamentals

This section of the paper contains a brief review of the manipulator dynamics and the intelligent techniques naming, RBNN and PSO.

2.1. Manipulator Dynamics:

The dynamics of revolute joint type of robot can be described by following nonlinear Lagrange equation (1) [22],

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau \quad (1)$$

with $q \in R^n$ as the joint position variables, τ as vector of input torques, $M(q)$ is the symmetric and positive definite inertia matrix, $V(q, \dot{q})$ is the coriolis and centripetal matrix, $G(q)$ includes the gravitational forces. Input torque given to the manipulator is of pivotal significance.

Manipulator used in this work is a *planar elbow manipulator with remotely driven link.* Unlike planar elbow manipulator, in this type of manipulator both the joints are driven by motors mounted at the base. The first joint is turned directly by one of the motors, while other is turned via a gearing mechanism or a timing belt as in Fig 1. Here, the generalized coordinates taken are as in Fig. 2, as the angle p_2 is determined by driving motor number 2 and is not affected by the angle p_1.

2.2 Radial Bias Neural Network (RBNN)

A typical RBNN consists of input layer, hidden layer and output layer as represented if Fig [3]. Input layer consists of input signals; hidden layer consists of radial bias functions (Gaussian function); output layer gives output by multiplying weights with the output of hidden layer. In this paper, input given to the RBNN is error and velocity error (e and $\dot{e}$) and output is obtained from NN is the input torque to be given to the manipulator for trajectory tracking control purpose.

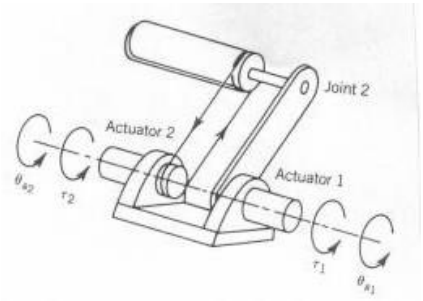

Fig. 1: Two link revolute joint arm with remotely driven link

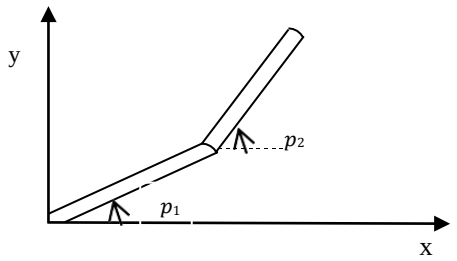

Fig. 2: Generalized coordinates for planar elbow manipulator with remotely driven links

The excitation values of this Gaussian function are distributed between the input values. The output of the hidden layer is given by equation (2) as

$$u = \sum_{j=1}^{n} w_j \exp\left[\frac{-\|s-c_j\|^2}{\sigma_j^2}\right] \quad (2)$$

where j is the j^{th} neuron of the hidden layer,
c_j is the central position of the neuron j,
σ_j is the spread factor of Gaussian function.

In output layer, output vector is given by $y = [\tau_1\ \tau_2]^T$ which vectorily can be written as the output of k^{th} neuron is given by equation (3)

$$y_k = \sum_{j=1}^{n} w_{jk} * u_k,$$
$$k\ 1,2 \ldots . \text{number of hidden layer neurons} \quad (3)$$

where w_{jk} represents the linking weight of the neuron in the output layer.

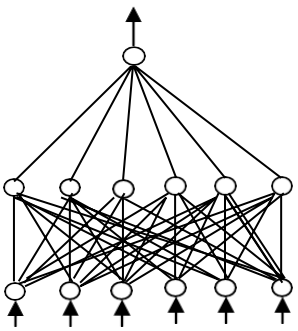

Fig. 3: RBNN architecture

Significance of the RBNN parameters to be optimized:

This section covers a brief discussion about the significance of the spread factor ( ) and the output weights (w_{jk}) in RBNN, followed by a discussion on the proposed control scheme.

a. Spread factor ( ) is the first parameter to be optimized using PSO. Spread factor () is of vital significance in RBNN. Its too small value can result in a solution that does not generalize from the input/target vectors and with a large value of it, the radial basis neurons will output large values (near 1.0) for all the inputs used to design the network. If radial basis neurons always output 1, any information presented to the network as input becomes lost. Hence, it is required to choose spread factor larger than the distance between adjacent

input vectors, so as to get good generalization, but smaller than the distance across the whole input space. It can be assumed that, it is crucial to have accurate results with the optimal value of this spread factor

b. Another performance deciding factor of RBNN is the selection of output weights (w_{jk}). Generally, these weights from hidden to output layer are decided by Least Square (LS) estimation [23]. In RBNN, these output weights could be affected by very commonly occurring noise and outliers in a nonlinear function. Hence, the approximation precision of RBNN could be consequently damaged with the presence of this external noise and outliers in the data set. Hence, it is always required to use some optimization technique to get the values of these weights for the improvements in the results and accuracy of NN based controllers.

2.3 Particle Swarm Intelligence (PSO)

In PSO starting with random population in search space, it results in the optimal solution. During each step every particle is accelerated towards its best neighboring position as well as in the direction of global best position. Calculation of new position of the swarm is given by equations (4) & (5) [12].

$$v_{id} = v_{id} + c_1 \in_1 (p_{id} - x_{id}) + c_2 \in_2 (p_{xd} - x_{id}) \quad (4)$$
$$x_{id} = x_{id} + v_{id} \quad (5)$$

where, in a D-dimensional space $\bar{x}_i = (x_{i1}, x_{i2}, \ldots x_{iD})$ is a present position vector, $\bar{p}_i = (p_{i1}, p_{i2}, \ldots p_{iD})$ is a best position vector, $\bar{v} = (v_{i1}, v_{i2}, \ldots v_{iD})$ is a velocity vector, , c_1 and c_2 are constant acceleration coefficients 2, $\in_1$ and $\in_2$ are the random number generators. In [24, 25] it has been proved that PSO finds the global best solution. PSO is becoming popular due to its simplicity in implementation and ability to converge quickly to a reasonably good solution.

Adaptive Weights in PSO

Although PSO is a new efficient emerging algorithm to the family of evolutionary algorithms and proven to be better than many other classical evolutionary techniques available (like Genetic Algorithm (GA)), yet there lies a huge scope for multi dimensional improvement in the basic PSO algorithm. One such improvement is made by incorporating a weight parameter on the previous velocity of the particle. The resulting equations for the manipulation of the swarm are [26] given in equations (6) & (7)

$$v_{id} = w * v_{id} + c_1 \in_1 (p_{id} - x_{id}) + c_2 \in_2 (p_{xd} - x_{id}) \quad (6)$$
$$x_{id} = x_{id} + v_{id} \quad (7)$$

where w is the inertia weight which manipulates the effects of the previous velocities on the current velocity. It can be said that w resolves the tradeoff between the global and the local exploration ability of the swarm. Literature reveals that w should have greater value in starting and should decrease gradually with iterations. As suggested by Hou in 2008[27] w adjusted adaptively proves itself as given in equation (8).

$$w = \frac{a}{b + [1g * iter]^2} \quad (8)$$

where $a = 0.6$, $b = 1$, iter is the current iteration.

This proposed adaptive weight in PSO has been applied to the manipulator of a planar robot with remotely driven links for the first time. Here, in this work i.e. for trajectory tracking control of robotic manipulator, this adaptive PSO has proven itself.

2.4 Friction Modeling

Friction forces between two surfaces in contact arises as a consequence of the irregularities and asperities at microscopical level, and their effects depend on many factors, such as displacement and relative velocity of bodies, properties of the surface materials, presence of lubrication, temperature etc. The experimental observation of friction phenomenon has led to various, deeply different models, which capture the friction component in a more or less accurate way. Friction is very important for the control engineer. Friction should be as much as reduced by good hardware design. But, with the advancements in the computers, computer control has also shown the possibility to reduce the effects of friction. This has been made possible using various mathematical friction modes. Interesting reviews of the main friction characteristics and classical models starting from the basic concept of friction as a force that opposes motion, captured by pure Coloumb model, up to complex static and dynamic models like LuGre friction model has been provided in literature. As opposed to classical static friction model, dynamic friction models attempt to incorporate a variety of other friction characteristics such as stiction, zero slip displacement, stribeck effect etc. Dynamic friction models also tend to capture effectively the changing friction characteristics that are caused primarily due to wear and aging. One of the most accurate dynamic frictions proposed is LuGre friction model. LuGre Fiction can be modeled mathematically as in equations (9)

$$\begin{aligned} F &= \sigma_o z + \sigma_1 z^{\cdot} + \sigma_2 v \\ z^{\cdot} &= v - \frac{|v|}{g(v)} z \\ g(v) &= F_c + (F_s - F_c) \exp\left[-\left(\frac{v}{v_s}\right)^2\right] \end{aligned} \quad (9)$$

where z is average bristle deflection, σ_o is stiffness of bristles, σ_1 is bristle damping coefficient, σ_2 is viscous damping coefficient, v is relative velocity between moving parts, F_c is coulomb coefficient, F_s is static coefficient, v_s is striberk velocity.

3. Proposed Controller

Even input output mapping in NN can be made by one of the many possible mapping functions yet the key issue in RBNN is not the selection of non-linear function but the key factor is the selection of constant parameters of these non-linear functions. Improper selection of some of the factors of RBNN can lead to unsatisfactory control results from RBNN. Spread factor (σ_j) and the network output weights (w_{jk}) are the few most performance deciding factors of RBNN. In other words, it can be said that proper selection of spread factor (σ_j) and the network output weights (w_{jk}) can be adjusted using one of the upcoming latest swarm intelligent technique naming PSO.

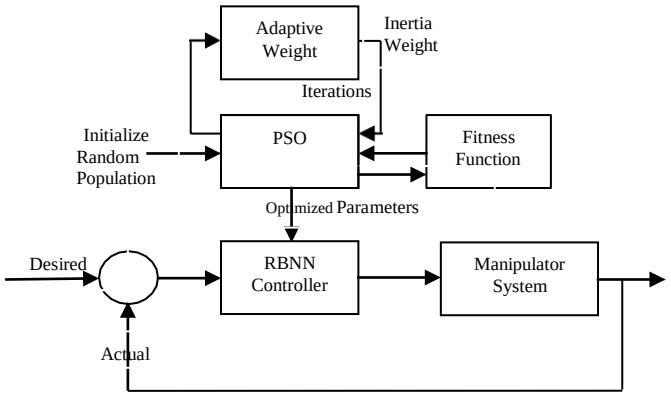

Fig. 4: Working of the proposed control scheme

In Fig. 4, it can be seen that PSO is used to obtain the optimized values of the RBNN parameters. PSO is initialized using a random population. Adaptive inertia weight in PSO has a different value for each iteration and hence, changes to adapt itself to the running PSO. As fitness function is a part of the basic PSO algorithm hence, it is evaluated for each iteration. Output of this PSO (optimized spread factor and output weights) is provided to RBNN.

Table 1: PSO Parameters

Population size	20
Number of Iterations	50
Inertia Weight (w)	2
Acceleration factors (c_1,c_2)	2
Fitness function	Root mean square of tracking error (RMSE)

This RBNN (with optimized constant parameters) is used to find the control input torque to be given to the manipulator for trajectory tracking. Tracking error and velocity tracking error are the inputs to the RBNN to have the control torque (given to the system to be controlled) as output. To have the values of error and velocity error actual trajectory tracked is compared to the desired trajectory. Flowchart representing the working of the control system is given in Fig. 5.

4. Simulation Example and Results

For the verification of the proposed controller, in this section a simulation study has been carried out. Control for a 2 DOF planar elbow with remotely driven links has been using the proposed controller has been implemented here. Dynamic model of the manipulator has been given in equations (10), (11) [22]

$$d_{11}\ddot{p}_1 + d_{12}\ddot{p}_2 + c_{221}\dot{p}_2^2 + \emptyset_1 = \tau_1 \quad (10)$$

$$d_{21}\ddot{p}_1 + d_{22}\ddot{p}_2 + c_{112}\dot{p}_2^2 + \emptyset_2 = \tau_2 \quad (11)$$

where

$d_{11} = m_1 l_{c1}^2 + m_2 l_1^2 + I_1$
$d_{12} = m_2 l_1 l_{c2} \cos(p_2 - p_1)$
$d_{21} = m_2 l_1 l_{c2} \cos(p_2 - p_1)$
$d_{22} = m_2 l_{c2}^2 + I_2$
$c_{221} = -m_2 l_1 l_{c2} \sin(p_2 - p_1)$
$c_{112} = m_2 l_1 l_{c2} \sin(p_2 - p_1)$
$g_1 = (m_1 l_{c1} + m_2 l_1) g \cos(p_1)$
$\emptyset_2 = m_2 l_{c2} g \cos(p_2)$

subscripts 1 & 2 indicates the link 1 & link 2; p_i is the angle with respect to horizontal axis; m_i is the weight; I_i is the inertia; l_i is the total length; l_{ci} is the distance from the joint to the centre of gravity; g is gravitational constant; τ_i is the torque input; where i is 1 & 2 for link 1 & link 2. Parameters for the manipulator taken for trajectory tracking are:

$m_1 = 10$; $m_2 = 5$; $I_1 = 0.2$; $I_2 = 0.2$; $l_{c1} = 0.25$; $l_{c2} = 0.5$; $g = 9.8$.

This manipulator is made to track the path for a two-link manipulator given by equation (12)

$$q_1 = \sin(0.67t) + \sin(0.3t) \quad (12a)$$
$$q_2 = \sin(0.39t) + \sin(0.5t) \quad (12b)$$

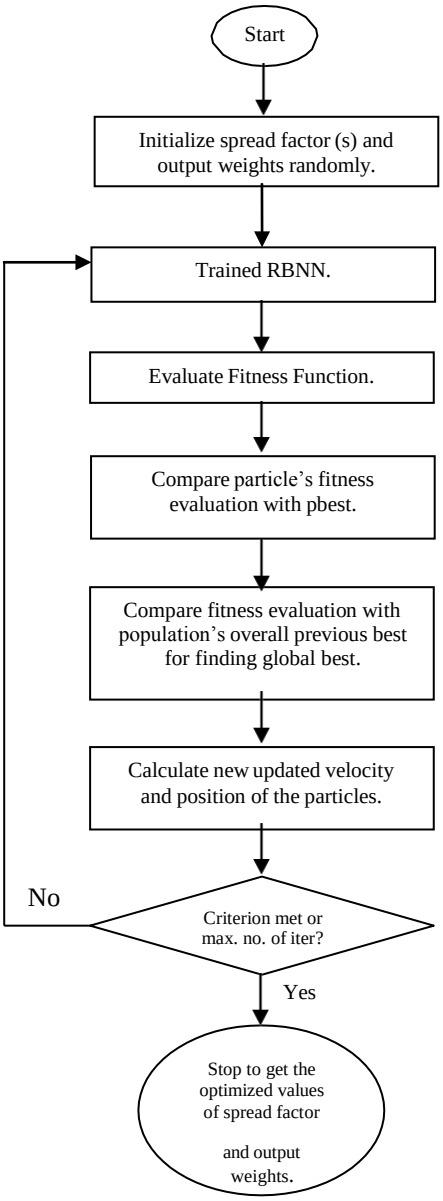

Fig. 5: Flowchart representing control scheme for the proposed controller

In this simulation study, for the trajectory tracking problem of planar manipulator, various controllers discussed in this paper have been implemented and the results have been compared. For payload changes ($m_2 + \Delta m$) is taken as 1.35 kg i.e. 35 % rise in

the mass of joint 2 of the manipulator. Parameters for LuGre friction model are chosen as:

$\sigma_o = 0.6, \sigma_1 = 0.009, \sigma_2 = 0.6, F_s = 0.01, F_c = 10$

4.1 RBNN Controller

For the implementation of the RBNN controller, given in Section II, the spread factor has been tuned manually between 0-2. The performance of the RBNN has been found best with spread factor as 2.

4.2 PSO Controller

Parameters chosen for a basic PSO controller have been given in Table 1 and results have been compiled in Fig. [6-9].

4.3 Proposed Controller

In proposed controller parameters for PSO are given in table 1 except the weight factor w, which is the adaptive in nature and is given in (8). Search range for spread factor has been taken as (0-2) and range for output weight factors has been taken as (0-1). Results for the trajectory tracking by the manipulator have been presented Figs. [6-9]. Although the graphs in Figs. [6-7] presents that the trajectory tracked by the manipulator using different control schemes is very close to each other, but the control performance of controllers can be easily differentiated with the help of tracking error graphs plotted in Figs. [8-9]. These graphs clearly represent that the best tracking performance is given by the proposed controller. Tracking errors of various controllers are given in Figs. [8-9]. Table 2 contains the performance indices to evaluate the performance of the controllers with all the uncertainties in terms of mean and mean square error (mse). Other type of error measuring performance indices like 2-norm error, integral square error (ISE) can also be evaluated and are found to show the similar type of results. It has been observed from table 2 that the max and mean error in Joint 1 & 2 is about 100 times lesser that the max and mean error of the other controllers. In joint 1, it can be observed that the mse is about 104 times lesser than the mse in other two control schemes whereas in joint 2 mse in the proposed controller is about 103 times lesser than the mse in other two control schemes. Hence, along with the robustness in the proposed control scheme, there is a rise in the accuracy in the tracking performance of the system under study.

Execution time (in seconds) for each control technique has been tabulated in table 3. It has been observed that RBNN is taking the maximum time for control execution. Proposed controller, along with less tracking error, is implementing the control action in lesser time when compared with RBNN.

5. Conclusion

As said, it would be safe here to infer again that the most commonly and widely used neural networks (NN) are not flawless, rather they have various shortcomings of their own including the dependency on experts for tunings its parameters, such as spread factor and output weights for good accuracy in results. This need is fulfilled by the proposed controller which uses one of the most emerging optimization techniques named as particle swarm optimization (PSO) to get the optimized parameters of RBNN for enhanced performance. This PSO enhanced RBNN controller has proved itself with accuracy in trajectory tracking. This controller also converges itself in lesser time as compared to a simple RBNN controller. Hence, as the outcome of the paper, it can be said that with the proposed robust control scheme perfect trajectory tracking problem of robotic manipulator has been solved upto a mark.

The study opens new vistas and futuristic avenues for further study, the more advanced and upgraded versions of PSO may be used for optimizing RBNN.

References

[1]. Lewis F. L., Jagannathan S. , and Yesildirek A.: *Neural Network Control of Robot Manipulators and Nonlinear Systems*. In: Taylor & Francis, 1998.

[2]. Benoudjit N., and Verleysen M.: *On the kernel widths in radial basis function networks.* In: Neural Processing Letters, Vol. 18 No. 2, 2003, pp.139-154.

[3]. M. Bernard, "Applying radial basis functions," *IEEE Signal Processing Magazine*, 13(2), 1996, pp. 50-65.

[4]. C. Panchapakesan, M. Palaniswami, and D. Ralph, "Effects of moving the center's in an RBF network", *IEEE Trans. Neural Networks*, 13(6), 2002, pp. 1299-1307.

[5]. L. P. Wang, and X. J. Fu, "Data Mining with Computational Intelligence", Springer, Berlin, 2005.

[6]. S. M. Bohte, H. La Poutre, and J. N. Kok, "Unsupervised clustering with spiking neurons by sparse temporal coding and multilayer RBF networks", *IEEE Trans. Neural Networks*, 13 (2), 2002, pp. 426-435.

[7]. X. J. Fu, and L P Wang, "Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance", *IEEE Trans. System, Man, Cybern, Part B-Cybernetics*, 33(3), 2003, pp. 399-409.

[8]. J. X. Peng, K. Li, and G W. Irwin, "A Novel Continuous Forward Algorithm for RBF Neural Modelling", *IEEE Trans. Automatic Control*, 52(1), 2007, pp. 117-122.

[9]. D.S. Broomhead, and D. Lowe, "Multivariable functional interpolation and adaptive networks", *Complex Systems 2*, 1988, pp. 321–355.

[10]. D. Lowe, "Adaptive radial basis function nonlinearities, and the problem of generalization", *Proceedings of IEE International Conference on Artificial Neural Networks*, 1989, pp. 171–175.

[11]. J.A.S. Freeman, and D. Saad, "Learning and generalization in radial basis function networks", *Neural Computation 9*, 1995, pp. 1601–1622.

[12]. J. Kennedy and R. Eberhart,, "Particle Swarm Optimization", *in Pro. IEEE Int. Conf. Neural Networks*,1995, pp. 1942- 1948.

[13]. C. Sudheer, R. Maheswaran, B.K. Panigrahi, and Shashi Mathur, "A Hybrid SVM-PSO Model for Forecasting Monthly Streamflow", *Neural Comput. & Applic.*, Feb, 2013.

[14]. The Berkeley Institute in Soft Computing. [Online]. Available: http://www-bisc.cs.berkeley.edu.

[15]. X. Yao, "Evolutionary Artificial Neural Networks", *Int. J. of Neural Systems*, 4(3), 1993, pp. 539-567.

[16]. H. Muhlenbein, "Limitations of Multi-Layer Preceptron Networks- Steps Towards Genetic Neural Networks", *Parallel Computing*, vol. 14, 1990, pp. 249-260.

[17]. S. Baluja, "Evolution of Artificial Neural Network Based Autonomous Land Vehicle Controller", *IEEE Trans. on SMC*, vol. 26, 1996, pp.450-463.

[18]. C. Longhan, L. Xiaoli, and G. Xiaodong, "The Application of Rought Set and Improved QPSO-RBF Algorithm to Fault Diagnosis for Diesel Engine Valve", *Information and Control*, 40 (4), 2011, pp. 570-576.

[19]. S. Xian, Z. Wen-guang, and Z. Yan, "Application of RBF Neural Network Based on PSO Algorithm in Fault Diagnosis of Actuation System," Journal of Naval Aeronautical and Astronautically University, 26 (2), 2011, pp. 131-135.

[20]. N. Kapoor, and J. Ohri, "A Neural Network Based Novel Approach for Error Optimization in Path Tracking Control of a Robotic Manipulator", *National Conference, AEMDS-2013, at TERII, Kurukshetra*, 2013, pp. 98-103.

[21]. N. Kapoor, and J. Ohri, "Fuzzy Sliding Mode Controller (FSMC) with Global Stabilization and Saturation Function for Tracking Control of a Robotic Manipulator", *Journal of Control and Systems Engineering*, Vol. 1 Iss. 2, Sept. 2013, pp. 50-56.

[22]. M. W. Spong, and M. Vidyasagar, "Robot Dynamics and Control. Wiley-India Edition", New York.

[23]. J. Park, and I.W. Sandberg, "Universal approximation using radial-basis-function networks", *Neural Computation 3*, 1991, pp. 246–257.

[24]. M. Clerc, "The Swarm and the Queen: Toward a Deterministic and Adaptive Particle Swarm Optimization", *Proc. IEEE Int. Congr. Evolutionary Computation*, vol. 3, 1999, p. 1957.

[25]. M. Clerc, and J. Kennedy, "The Particle Swarm-Explosion, Stability and Convergence in a Multi-Dimensional Complex Space", *IEEE Trans. Evol. Comput.*, vol. 6, Feb. 2002, pp. 58-73.

[26]. J. Kennedy and R. Eberhart, Swarm Intelligence, *Mergan Kaufmann Publishers*, 2001.

[27]. X. Hou, "Wiener model identification based on adaptive particle swarm optimization", *IEEE Proceedings of Seventh International Conference Machine Learning and Cybernatics*, Kumming 12–15th July, 2008,pp. 1041–1045.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflict of interest to declare that is relevant to the content of this article.

Table 2: Tracking Errors: Joint 1 & 2

Uncertainties	Control Scheme	Joint 1			Joint 2		
		max. abs. error	Mean error	mse	max. abs. error	Mean error	mse
Payload changes	RBNN	0.0494	0.0326	0.0013	0.0686	0.0337	0.0014
	PSO	0.0613	2.15e-02	7.51e-04	0.0545	2.24e-02	7.00e-04
	Proposed	0.002	0.0018	3.32e-06	4.13e-04	2.56e-04	6.69e-08
LuGre Friction	RBNN	0.116	0.0318	0.002	1.03e-01	0.0306	0.0018
	PSO	0.0697	0.0199	7.57e-04	0.0587	0.0173	5.70e-04
	Proposed	2.53e-04	7.80e-05	1.75e-08	2.92e-04	1.77e-04	3.44e-08
Both	RBNN	0.0587	0.0257	9.18e-04	0.0618	0.0248	9.15e-04
	PSO	0.0637	0.0236	8.35e-04	0.0505	0.0197	5.72e-04
	Proposed	4.31e-04	2.55e-04	7.58e-08	4.52e-04	3.81e-04	1.47e-07

Table 3: Control execution time (in seconds)

Controllers	**RBNN**	**PSO**	**Proposed Controller**
time	101.99	28.77	48.34

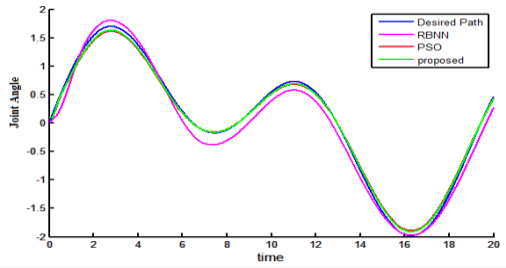

Fig. 6: Trajectory tracking response by Joint 1

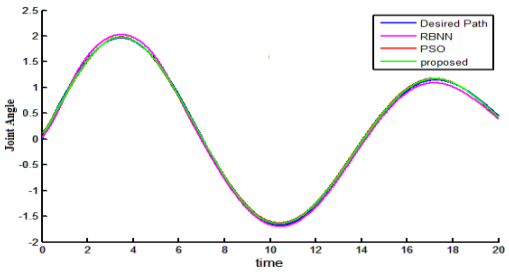

Fig. 7: Trajectory tracking response by Joint 2

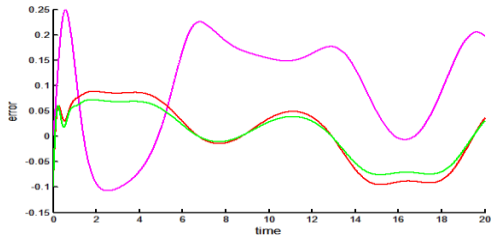

Fig. 8: Tracking error for Joint 1

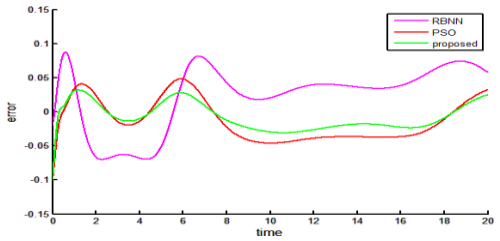

Fig. 9: Tracking error for Joint 2