Hybrid Neural Network Based Prediction of Inverse Kinematics of Robot Manipulator

Panchanand Jha[1*] and B. B. Biswal[2]

[1*,2]Department of Industrial Design, NIT Rourkela, and PIN: 769008,

E-mail: jha_ip007@hotmail.com; bbbiswal@nitrkl.ac.in;

Abstract

The fundamental of the inverse kinematics of robot manipulator is to determine the joint variables for a given Cartesian position and orientation of an end effector. Conventional methods to solve inverse kinematics such as geometric, iterative and algebraic are complex for redundant manipulators. There is no unique solution for the inverse kinematics thus necessitating application of appropriate predictive models from the soft computing domain. Although artificial neural network (ANN) can be gainfully used to yield the desired results, but the gradient descent learning algorithm does not have ability to search for global optimum and it gives a slow convergence rate. This paper proposes structuring ANN with hybridization of Particle Swarm Optimization to solve the inverse kinematics of 6R robot manipulator. An investigation has been made on accuracies of adopted algorithm. The ANN model used is multi-layered perceptron neural network (MLPNN) with back-propagation (BP) algorithm which is compared with hybrid multi layered perceptron particle swarm optimization (MLPPSO). An attempt has been made to find the best ANN configuration for the problem. It has been observed that MLPPSO gives a faster convergence rate and improves the problem of trapping in local minima. It is found that MLPPSO gives better result and minimum error as compared to MLPBP.

Keywords:Inverse kinematics, D-H algorithm, PSO, MLP

1 Introduction

An industrial robot consists of a set of rigid links connected together by a set of joints. To control the overall motion of a mechanism for each links connected by various joints like revolute or prismatic is performed by motors. Generally tool or end effector performs tasks in the Cartesian coordinate system which is controlled by joint coordinate system. For better position and orientation of robot end effector to perform the stated task, it is essential to understand the kinematics relationship between the joint coordinate system and the Cartesian coordinate system.

Generally there are two types of kinematic analysis, which is forward kinematics and inverse kinematics. Forward kinematics is a conversion of joint space variables into end-effector position and orientation. Conversion of the position and orientation of robot manipulator end-effectors from Cartesian space to joint space is called as an inverse kinematics problem. This is of fundamental importance in calculating desired joint angles for robot manipulator design and positioning. The corresponding joint values must be computed at high speed by the inverse kinematics transformation Xu et al. (2005). For a manipulator with no degree of freedom, at any instant of time joint variable is denoted by $\theta i = \theta(t)$, I = 1, 2,3..... n and position variables by $x_j = x(t)$, j = 1, 2, 3..... m. The relations between the end-effectors position x (t) and joint angle θ (*t*) can be represented by the forward kinematic equation

$$x(t) = f(\theta(t)) \quad (1)$$

where, f is a nonlinear continuous and differentiable function. On the other hand, with the desired end effectors position, the problem of finding the values of the joint variables is inverse kinematics, which can be solved by,

$$\theta(t) = f^{'}(x(t)) \quad (2)$$

Inverse kinematics solution is not unique due to nonlinear, uncertain and time varying nature of the governing equations Chiddarwar and Babu (2010).The different techniques used for solving inverse kinematics can be classified as algebraic, geometric and iterativeAlavandar and Nigam (2008). The algebraic methods do not guarantee closed form solutions. In case of geometric methods, closed form solutions for the first three joints of the manipulator must exist geometricallyHusty et al. (2007). The iterative methods converge to only a single solution depending on the starting point and may not work near singularities. In case of numerical method the major difficulty of inverse kinematics is that, when the Jacobian matrix is singular or ill-conditioned, it does not find a solution. In addition, if the initial approximation of the solution vector (i.e. The vector of joint variables) is not sufficiently accurate, this method may become unstable Olaru and Olaru (2011). Because of the above mentioned reasons, various

authors adopted ANN.The simulation and computation of inverse kinematics using multilayer feed perceptron network is particularly useful where less computation times are needed, such as in real-time adaptive robot control Mirjalili et al. (2012). If the number of degrees of freedom increases, traditional methods will become more complex and quite difficult to solve inverse kinematics Zhang et al. (2007).

Although the use of ANN is not new in the field of multi-objective and NP-hard problem to arrive at a very reasonable optimized solution, the MLPPSO has not been tried to solve inverse kinematics problem for 6R PUMA robot manipulator. Therefore, the main aim of this work is focused on minimizing the mean square error of the neural network-based solution of the inverse kinematics problem using PSO. The training data of neural network have been selected very precisely. Especially, unlearned data in each neural network have been chosen, and used to obtain the training set of the last neural network.

2 Mathematical Modelling of 6R PUMA Manipulator

Denavit-Hartenberg (DH) algorithm is used to calculate the individual homogeneous transformation matrices which then use to derive the forward and inverse kinematics of 6R PUMA robot manipulator. DH parameters and associated values for PUMA manipulator have given in table 1 and assigned coordinate frames are shown in "Fig. 1,"

Table 1 D-H Parameters

Frame	θ_i (degree)	d_i (m)	a_i (m)	α_i (degree)
0	θ_1	0	0	0
1	θ_2	0	0	-90
2	θ_3	d_3=0.1244	a_1=0.4318	0
3	θ_4	d_4=0.4318	a_2=0.0203	-90
4	θ_5	0	0	90
5	θ_6	0	0	-90

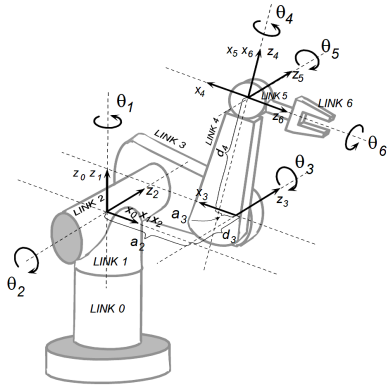


Figure 1 Model and coordinate frames of manipulator

Inverse kinematics of PUMA manipulator is given below:

$$\theta_1 = a\tan 2(\pm\sqrt{Px^2+Py^2-d_3^2}, d_3) - a\tan 2(Px, Py) \quad (3)$$

$$\begin{aligned}\theta_2 &= a\tan 2(-Pz, \pm\sqrt{Px^2+Py^2-d_3^2}) \\ &- a\tan 2(\pm\sqrt{a_3^2+d_4^2-b_2^2}, b_2+a_2)\end{aligned} \quad (4)$$

$$\text{where, } b_2 = \frac{p^2-a_2^2-a_3^2-d_3^2-d_4^2}{2a_2},$$

$$p = \sqrt{Px^2+Py^2+Pz^2}$$

θ_2 can also be expressed in other form:

$$\begin{aligned}\theta_2 &= a\tan 2(\pm\sqrt{Px^2+Py^2-d_3^2}, Pz,) \\ &- a\tan 2(\pm\sqrt{a_3^2+d_4^2-b_2^2}, b_2+a_2) - \frac{\pi}{2}\end{aligned} \quad (5)$$

$$\theta_3 = a\tan 2(\pm\sqrt{a_3^2+d_4^2-b_2^2}, b_2) - a\tan 2(d_4, a_3) \quad (6)$$

$$\theta_4 = \begin{cases} -\pi - a\tan 2(g_{23}, g_{13}), \text{ if } \sigma_3 < 0 \\ a\tan 2(g_{23}, g_{13}), \text{ if } \sigma_3 > 0 \end{cases} \quad (7)$$

$$\text{Where, } \sigma_3 = \pm\sqrt{g_{31}^2+g_{32}^2}$$

$$\theta_5 = -a\tan 2(\sigma_3, g_{33}) \quad (8)$$

$$\theta_6 = \begin{cases} -a\tan 2(g_{32}, g_{31}), \text{ if } \sigma_3 < 0 \\ \pi - a\tan 2(g_{32}, g_{31}), \text{ if } \sigma_3 > 0 \end{cases} \quad (9)$$

Where (p_x, p_y, p_z) represents the position and $\{(n_x, n_y, n_z), (o_x, o_y, o_z), (a_x, a_y, a_z)\}$ the orientation of the end-effector.

It is obvious from the equations (3) through (9) that there exist multiple solutions to the inverse kinematics problem. By comparing the errors between these four generated positions and orientations and the given position and orientation, one set of joint angles, which produces the minimum error, is chosen as the correct solution.

3 Application of PSO for training MLP

We propose the solution using a multi-layered perceptron with the back-propagation algorithms for training. The network is then trained with data for a number of end effector positions expressed in Cartesian co-ordinates and the corresponding joint angles. The data consist of the different configurations available for the arm. The different poses of the arm are then used to train a three-layer, fully connected back-propagation model shown in in "Fig. 2,".Each of the signals from the input neurons is multiplied by the value of the weights of the connection weights between the respective input neurons and the hidden neuron.

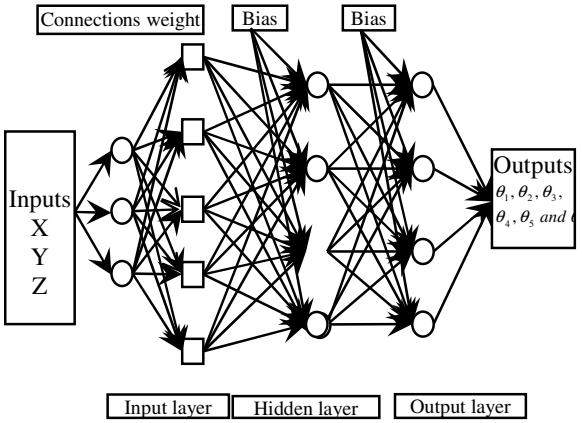


Figure 2: A block diagram of the system using ANN

Analytical solution of inverse kinematics problem is highly non-linear and mathematically complex in nature. An ANN model does not require higher initial selection of weight which is vigorous to yield local optima, convergence speed and training time for the network. Generally weight is randomly selected in the range of 0 to 1, after activation function weight of each neurons adjusted for the next iteration. The heuristic optimization algorithm optimizes the weights of the neural networks. When certain termination criteria are met, or a maximum number of iterations are reached, the iterations cease. From the previous research hybrid optimization algorithm started evolving with high and remarkable advances in their performances Kennedy and Eberhart (1995). These techniques produces better outflow from local optimum and testified to be more operative than the standard method. In this paper we have optimized weight and bias for each neuron using PSO as shown in "Fig. 3,". For the training of network it is important to have all connection weights and biases in order to minimize the mean square error. To optimize MLP neural network it is important to have fitness function PSO and then it is required to define the initial weight and bias for the training of MLP neural networkMirjalili (2012). The basic steps and flow chart of MLPPSO has given in "Fig.3,".

Learning error E (fitness function) is calculated from equation (10-11).

$$E_k = \sum_{i=1}^{m} (o_i^k - y_i^k)^2 \tag{10}$$

$$E = \sum_{k=1}^{q} (\frac{E_k}{q}) \tag{11}$$

where q is the number of training samples, y_i^k is the desired output of the ith input unit when the kth training sample is used, and o_i^k is the actual output of the ith input unit when the kth training sample is used.

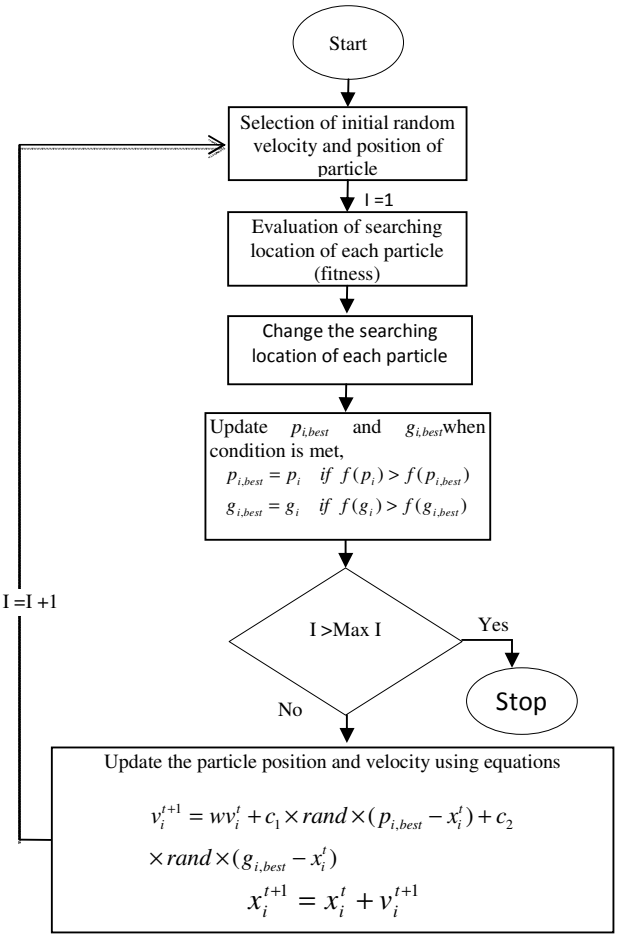


Figure 3 Flow chart for MLPPSO

Fitness function can be calculated from equation (12). Where the number of input nodes is equal to n, the number of hidden nodes is equal to h, and the number of output nodes is m. Therefore, the fitness function of the ith training sample can be defined as follows:

$$Fitness(X_i) = E(X_i) \tag{12}$$

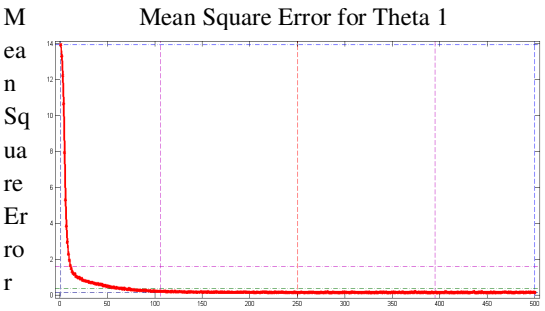
4 Results and Discussion

The proposed work is performed on the Matlab R2013a. Back-propagation algorithm was used for training the network and for updating the desired weights. In this work the training data sets were generated by using equation (3) through (15). A set of 1000 data sets were first generated as per the formula for the input parameter px, py and pz coordinates in mm. These data sets were the basis for the training, evaluation and testing the MLP model. The following parameters were taken: learning rate 0.36, momentum parameter 0.41, number of epoch 500, number of hidden layer 2, number of inputs 3 and number of output 6.

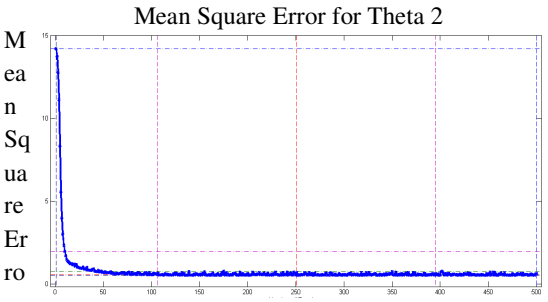
The MSE for MLPBP algorithm shown in "Fig. 3,"the used solution method gives the chance of selecting the output, which has the least error in the system. So, the solution can be obtained with less error. Table 2 gives the experimental results and comparison between the MLPBP algorithms with respect to hybrid MLPPSO for two hidden layers. "Fig. 3," (a), (b), (c), (d), (e) and (f) shows the selected best mean square curve of MLPBP for all joint variables. Similarly best chosen mean square curve of MLPPSO from table 2 depicted in "Fig. 4," (a), (b), (c), (d), (e) and (f) for all joint variables.

Table 2 Mean square error for all joint angles

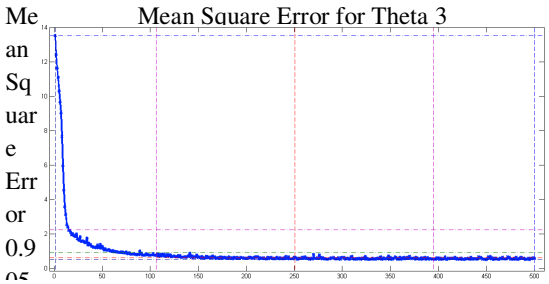
Sn.	Mean square error of MLPBP	Mean square of MLPPSO
1	0.3825	3.5476e-08
2	0.5862	3.7132e-07
3	0.9054	1.6987e-06
4	1.704e-2	2.2308e-07
5	0.5237	6.1225e-11
6	0.1434	7.7393e-09



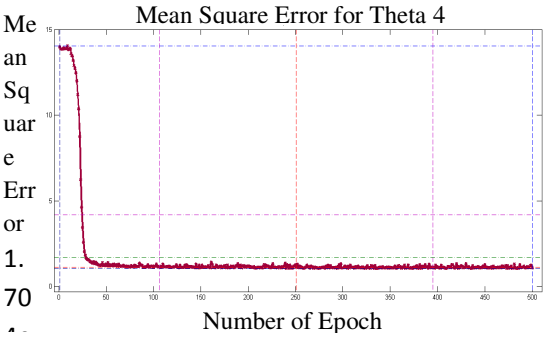
(a)



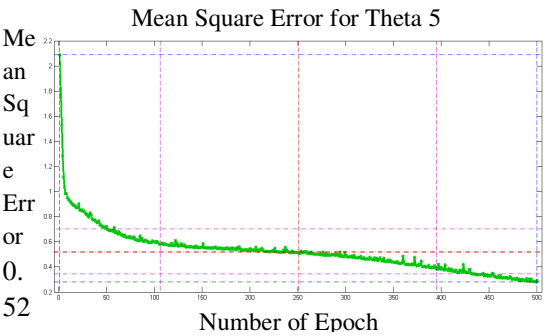
(b)



(c)



(d)



(e)

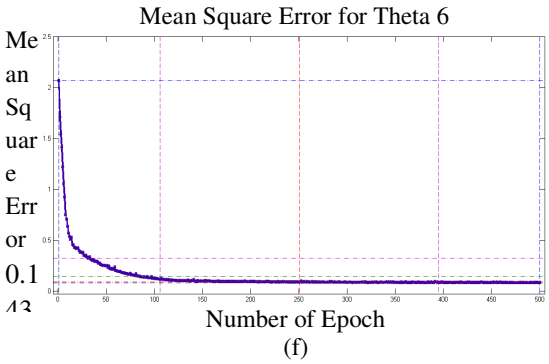


Figure 3 Figure (a), (b), (c), (d), (e) and (f) are mean square error curve of MLPBP for all joint angles.

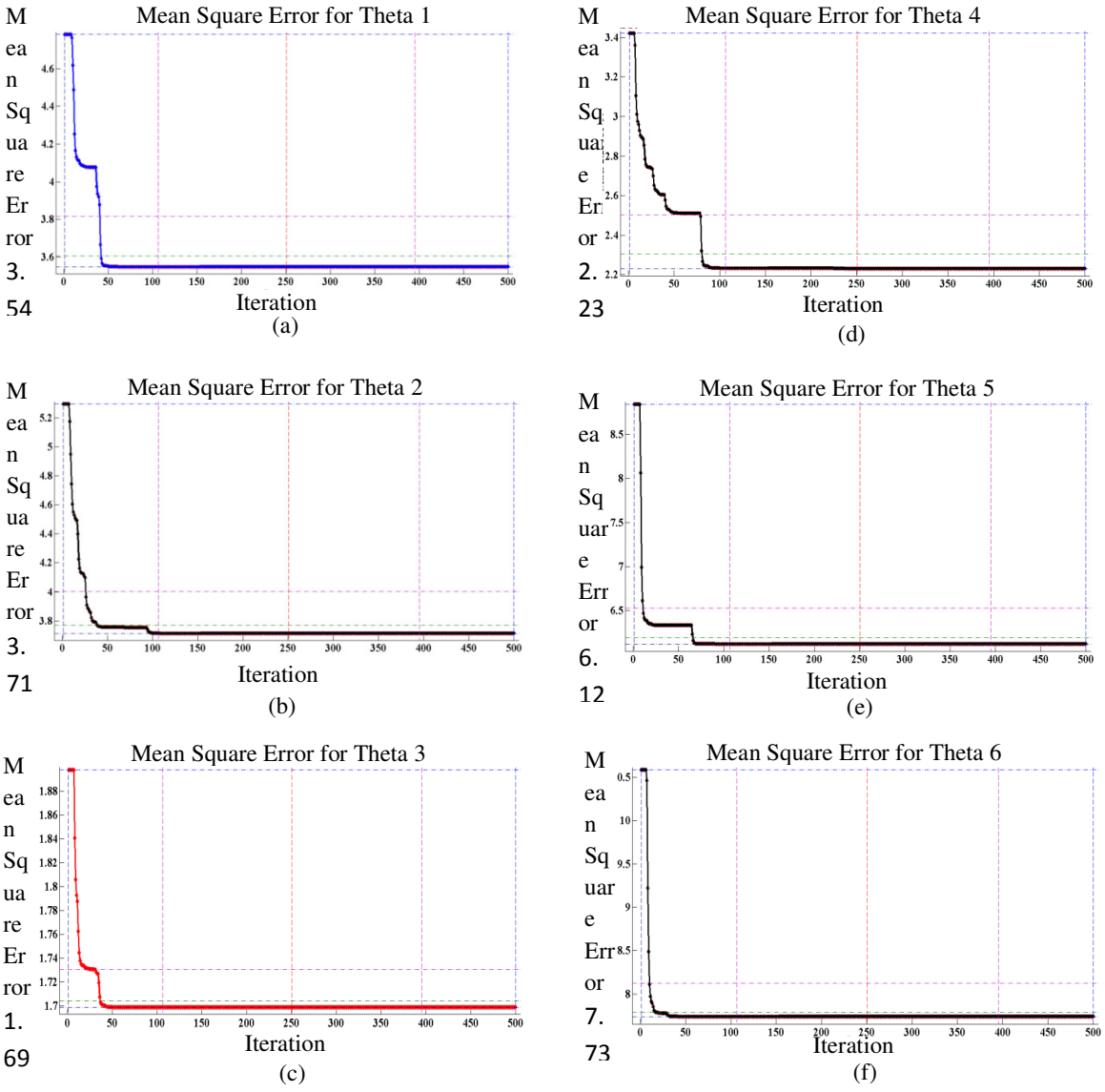


Figure 4Figure (a), (b), (c), (d), (e) and (f) are mean square error curve of MLPPSO for all joint angles.

5 Conclusions

In this paper, we have selected two methods which are MLPBP and MLPPSO to obtain the solution of inverse kinematics of 6R manipulator. In this approach forward and inverse kinematic model of 6R manipulator is used to generate the data set for training the MLP. The difference in desired and predicted data with MLPBP, gives poor results as compared to MLPPSO. Also, the MLPPSO accumulate small number of epoch with hybrid learning algorithm. Therefore, MLPPSO can be used for accurate and fast solution of inverse kinematics. Future research will revise the rules, inputs, number and type of membership functions, the epoch numbers used, and training sample to further refine the MLPPSO model.

References

Alavandar S. and Nigam M. J., (2008), Neuro-Fuzzy based Approach for Inverse Kinematics Solution of Industrial Robot Manipulators, *Int. J. of Computers, Communications & Control*, vol. 3,pp. 224-234.

ChiddarwarS. S and BabuN. R. (2010), Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach, *Engineering Applications of Artificial Intelligence*vol. 23, 1083–1092.

HustyM. L., PfurnerM. and SchrockerH. P., (2007), A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator, *Mechanism and Machine Theory* vol. 42, 66-81.

Kennedy J. and Eberhart R.C., (1995), Particle swarm optimization, in: *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948.

Mirjalili S., Hashim S. Z. M and Sardroudi,H. M. (2012), Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, *Applied Mathematics and Computation* vol. 218, 11125–11137.

OlaruA. and OlaruS. (2011), Optimization of the robots inverse kinematics results by using the Neural Network and LabView simulation, *IPCSIT*, vol.13.

Sarafrazi S., Nezamabadi-pour H. and Saryazdi S. (2011), Disruption: A new operator in gravitational search algorithm", *ScientiaIranica*, vol. 18 (3), 539–548.

Wasserman P.D., (1989), Neural Computing, Van Nostrand Reinhold, NewYork, 1989, 230 pages.

Xu D. et al., (2005), An Analysis of the Inverse Kinematics for a 5-DOF Manipulator, *International Journal of Automation and Computing*vol.2, 114-124.

Zhang J. R. et al., (2007), A hybrid particle swarm optimization–back-propagation algorithm for feed forward neural network training", *Applied Mathematics and Computation* vol. 185.1026–1037.