Inverse Kinematic Solution of Robot Manipulator Using Hybrid Neural Network

Panchanand Jha
National Institute of Technology, Department of Industrial Design, Rourkela, India
Email: jha_ip007@hotmail.com

Bibhuti B. Biswal and Om Prakash Sahu
National Institute of Technology, Department of Industrial Design, Rourkela, India
Email: {bbbiswal, 512id101}@nitrkl.ac.in

***Abstract*—Inverse kinematics of robot manipulator is to determine the joint variables for a given Cartesian position and orientation of an end effector. 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 slow convergence rate. This paper proposes structured ANN with hybridization of Gravitational Search Algorithm to solve inverse kinematics of 6R PUMA robot manipulator. The ANN model used is multi-layered perceptron neural network (MLPNN) with back-propagation (BP) algorithm which is compared with hybrid multi layered perceptron gravitational search algorithm (MLPGSA). An attempt has been made to find the best ANN configuration for the problem. It has been observed that MLPGSA gives faster convergence rate and improves the problem of trapping in local minima. It is found that MLPGSA gives better result and minimum error as compared to MLPBP.**

***Index Terms*—forward Kinematics, Inverse kinematics, multi-layered neural network, D-H parameters, gravitational search algorithm, back propagation algorithm**

I. Introduction

An industrial robot consists of a set of rigid links connected together by set of joints. To control the overall motion of 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 stated task, it is essential to understand 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 conversion of joint spaces 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 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 [1]. For a manipulator with n degree of freedom, at any instant of time joint variables 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 $\theta(t)$can be represented by forward kinematic equation

$$x(t) = f(\theta(t)) \tag{1}$$

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,

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

Inverse kinematics solution is not unique due to nonlinear, uncertain and time varying nature of the governing equations [2-3]. The different techniques used for solving inverse kinematics can be classified as algebraic, geometric and iterative. 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 geometrically. 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

Manuscript received xxxxxx 2014; revised xxxx 2014; accepted xxxx, 2014.

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 [4-5]. 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 [6-7]. If the number of degrees of freedom increases, traditional methods will become more complex and quite difficult to solve inverse kinematics [8].

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 MLPGSA 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 inverse kinematics problem using GSA. 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. The sectional organization of the paper henceforth is as follows: section 2 pertains to the mathematical modelling of 6R PUMA robot manipulator. Introduction to standard MLP and GSA have been briefed using flowcharts and associated equations in section 3 along with the method of applying GSA to ANN as evolutionary training algorithms. The experimental results as obtained from simulations are discussed elaborately in Section 4.

II. MATHEMATICAL MODELLING OF 6R PUMA ROBOT 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,"

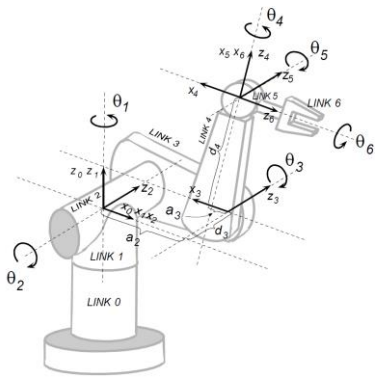


Figure 1. Model and coordinate frames of manipulator

TABLE I. 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	0	0	0	-90

Inverse kinematics of PUMA manipulator is given below:

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

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

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:

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

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

We can separate the arm and wrist if the manipulator has spherical wrist.

Therefor rotation matrix for arm can be given by:

$$R_A = \begin{bmatrix} c_1c_{23} & s_1 & -c_1s_{23} \\ s_1c_{23} & -c_1 & -s_1s_{23} \\ -s_{23} & 0 & -c_{23} \end{bmatrix} \quad (7)$$

Position matrix for arm can be given by:

$$P_A = \begin{bmatrix} Px \\ Py \\ Pz \end{bmatrix} = \begin{bmatrix} c_1(c_{23}a_3 - s_{23}d_4 + c_2a_2) - s_1d_3 \\ s_1(c_{23}a_3 - s_{23}d_4 + c_2a_2) + c_1d_3 \\ -s_{23}a_3 - c_{23}d_4 - s_2a_2 \end{bmatrix} \quad (8)$$

Now general equation for spherical wrist can be evaluated from mapping of z-y-z Euler angle into given rotation matrix:

$$\text{Z-Y-Z}\ (\theta_4, -\theta_5, \theta_6) = \text{G} \quad (9)$$

Where, $G = R_A^T R$

Therefore we can evaluate elements of matrix G from equation "(9),"

$$g_{1i} = (c_1c_{23})r_{1i} + (s_1c_{23})r_{2i} - s_{23}r_{3i} \quad (10)$$

$$g_{2i} = s_1 r_{1i} - c_1 r_{2i} \quad (11)$$

$$g_{3i} = -(c_1 s_{23}) r_{1i} - (s_1 s_{23}) r_{2i} - c_{23} r_{3i} \quad (12)$$

Therefore,

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

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

$$\theta_5 = -a\tan 2(\sigma_3, g_{33}) \quad (14)$$

$$\theta_6 = \begin{cases} -a\tan 2(g_{32}, g_{31}), \text{ if } \sigma_3 < 0 \\ \pi - a\tan 2(g_{32}, g_{31}), \text{ if } \sigma_3 > 0 \end{cases} \quad (15)$$

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 (15) 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.

III. APPLICATION OF GSA FOR TRAINING MLP

Analytical solution of inverse kinematics problem is highly non-linear and mathematically complex in nature. An ANN model does not require higher mathematical calculations and complex computing program. ANN requires 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 [9, 10]. 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 GSA as shown in "Fig. 2,". 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 GSA and then it is required to define the initial weight and bias for the training of MLP neural network. The basic steps and flow chart of MLPGSA has given in "Fig. 2,".

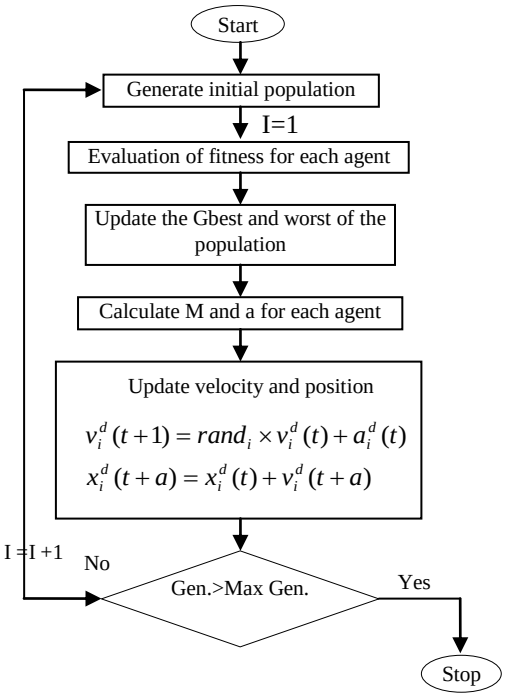


Figure 2. Flow chart for MLPGSA

(A) *fitness function*

From [5-7,9], in each epoch of learning, the output of each hidden node is calculated from equation (16).

$$f(n_k) = \frac{1}{(1 + \exp(-(\sum_{i=1}^{n} w_{ij}.x_i - b_j)))}, \quad j = 1,2,3,...,h \quad (16)$$

Where $n_k = \sum_{i=1}^{n} w_{ij}.x_i - b_j$,n is the number of the input nodes, w_{ij} is the connection weight from the ith node in the input layer to the jth node in the hidden layer, b_j is the bias (threshold) of the jth hidden node, and x_i is the ith input. After calculating outputs of the output nodes from equation (17).

$$o_k = \sum_{i=1}^{h} w_{kj}.f(n_k) - b_k, \quad k = 1,2,...,m \quad (17)$$

where w_{kj} is the connection weight from the jth hidden node to the kth output node and b_k is the bias (threshold) of the kth output node. Finally, the learning error E (fitness function) is calculated from equation (18-19).

$$E_k = \sum_{i=1}^{m} (o_i^k - y_i^k)^2 \quad (18)$$

$$E = \sum_{k=1}^{q} (\frac{E_k}{q}) \quad (19)$$

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. Fitness function can be calculated from equation (20). 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:

$$Fitness(X_i) = E(X_i) \tag{20}$$

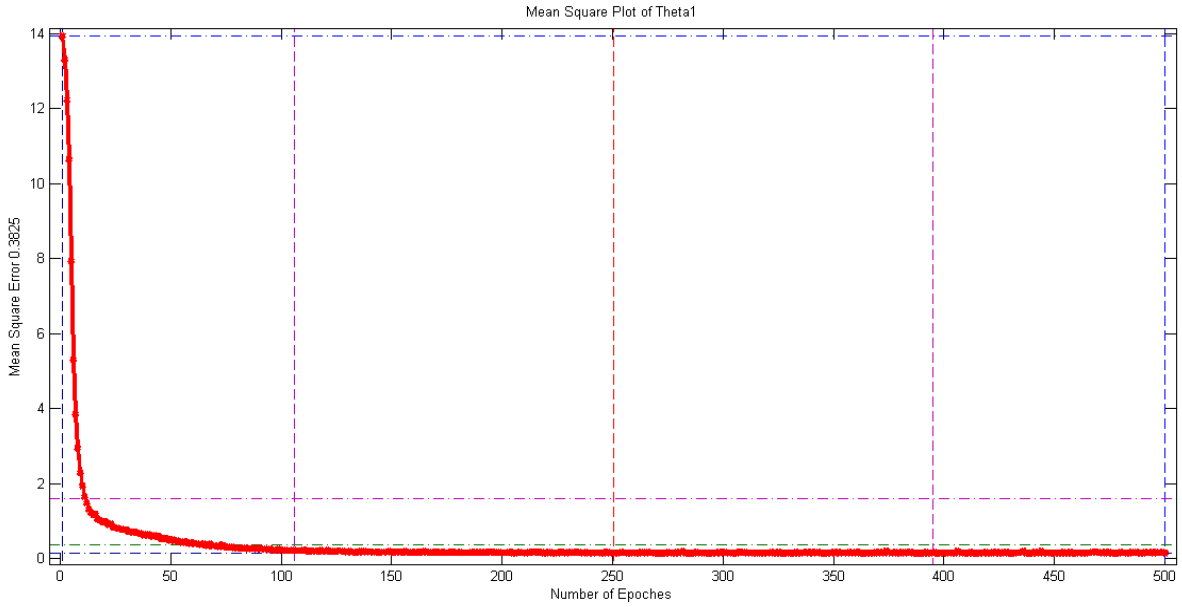
IV. RESLUTS AND DISCUSSION

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 (17). 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 0.45, momentum parameter 0.86, number of epoch 500, number of hidden layer 2, number of inputs 3 and number of output 6.

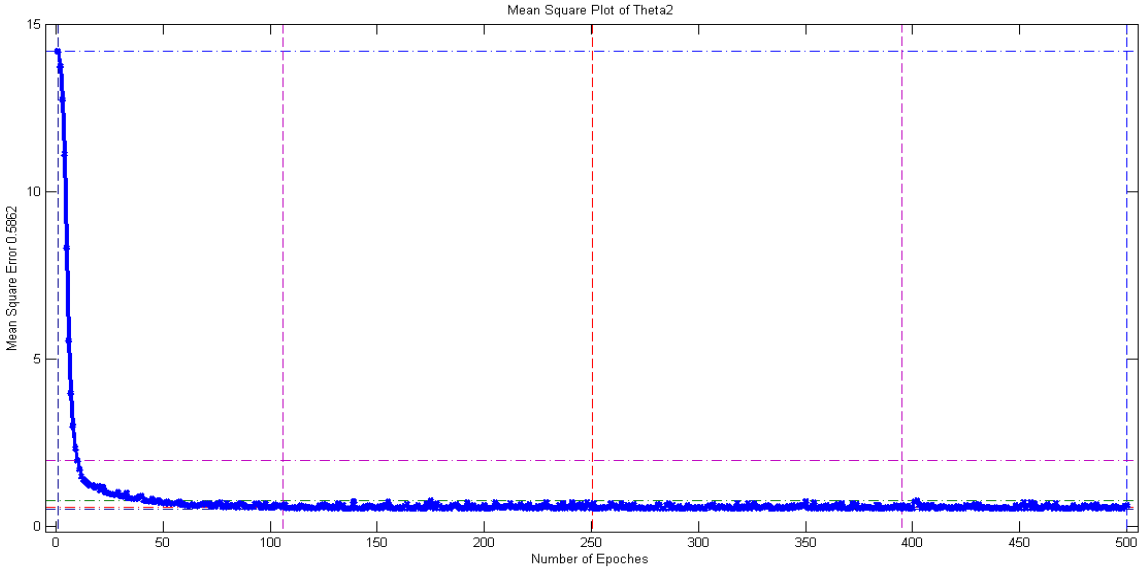
The MSE for MLPBP algorithm shown in "Fig. 3," the used solution method gives the chance of selecting the output, which has the least error in the system. So, the solution can be obtained with less error. Table 3 gives the experimental results and comparison between the MLPBP algorithms with respect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

TABLE II. MEAN SQUARE ERROR FOR ALL JOINT ANGLES FOR MLPBP AND MLPGSA

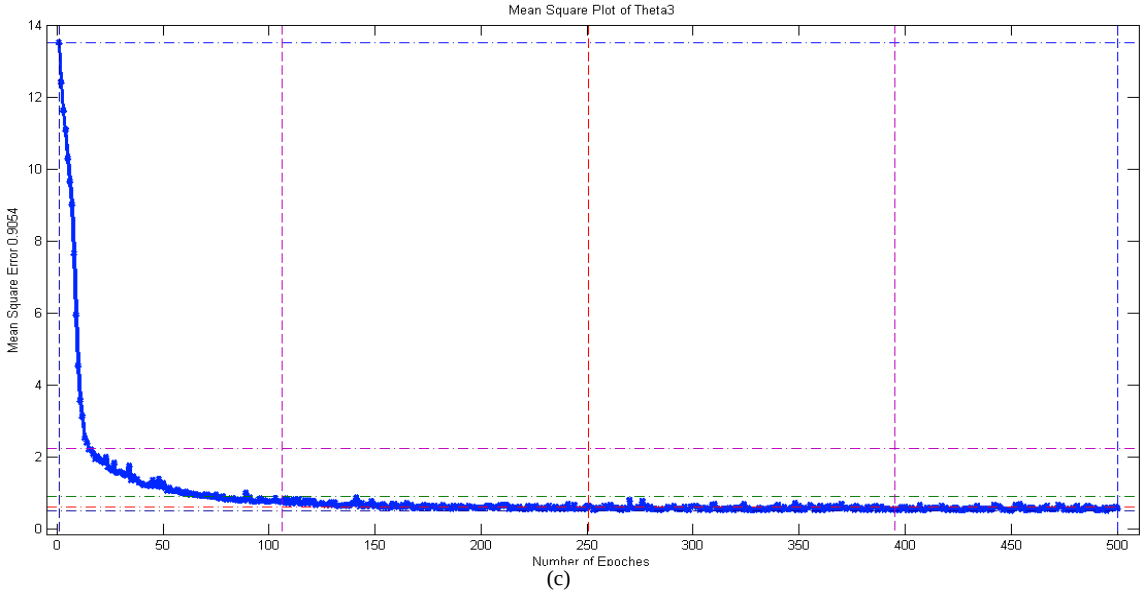
Sn.	Mean square error of MLPBP	Mean square of MLPGSA
1	0.3825	2.7571e-10
2	0.5862	1.7161e-12
3	0.9054	1.9841e-14
4	1.704e-2	9.5988e-17
5	0.5237	2.6961e-16
6	0.1434	8.4772e-15



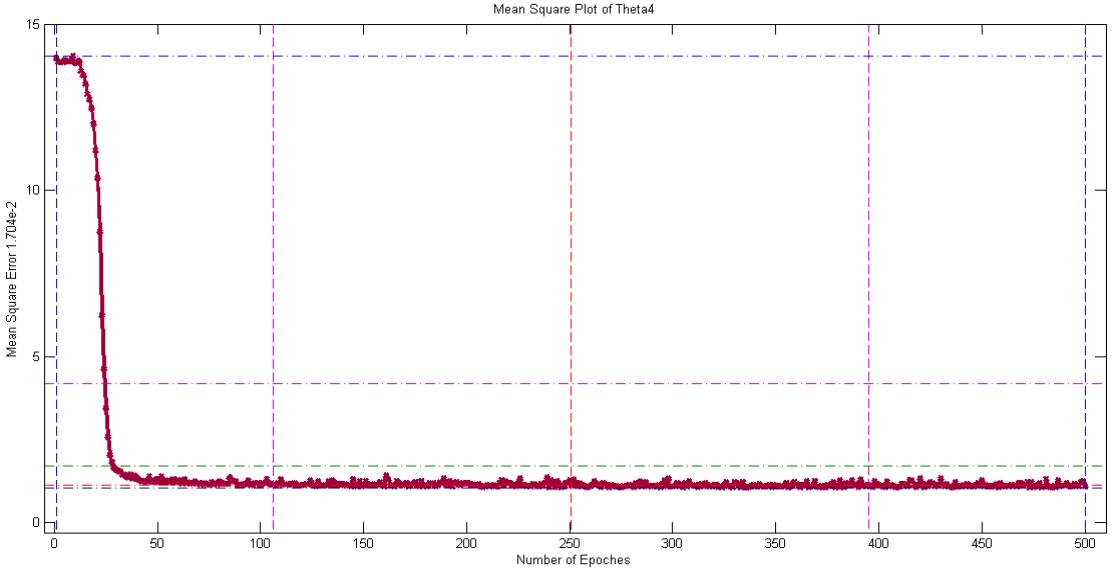
(a)


Mean Square Plot of Theta2
Mean Square Error 0.5862
Number of Epoches

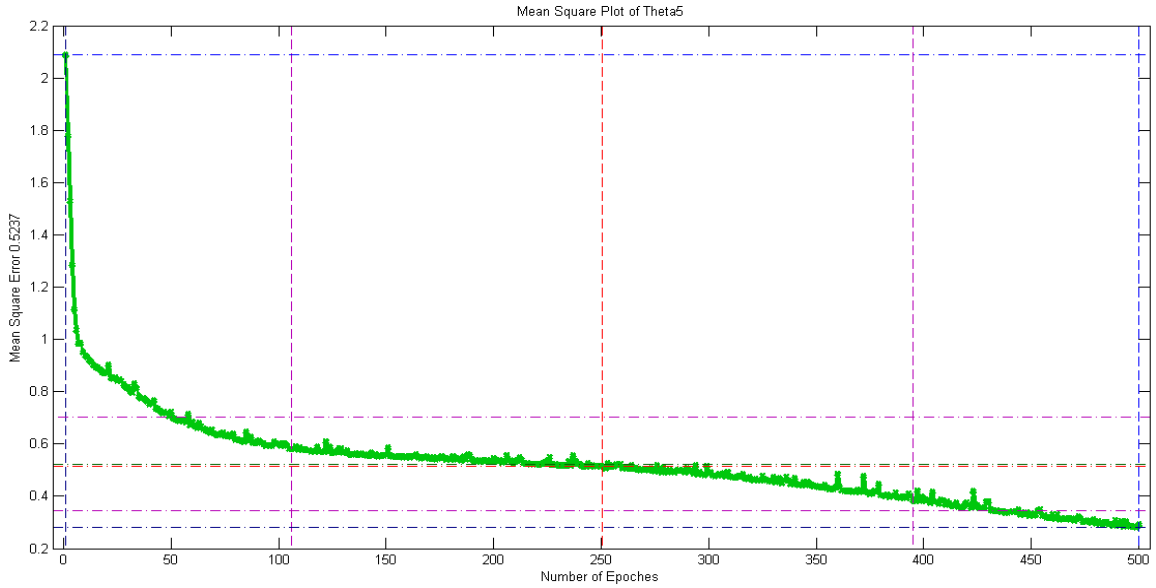
(b)


Mean Square Plot of Theta3
Mean Square Error 0.9054
Number of Epoches

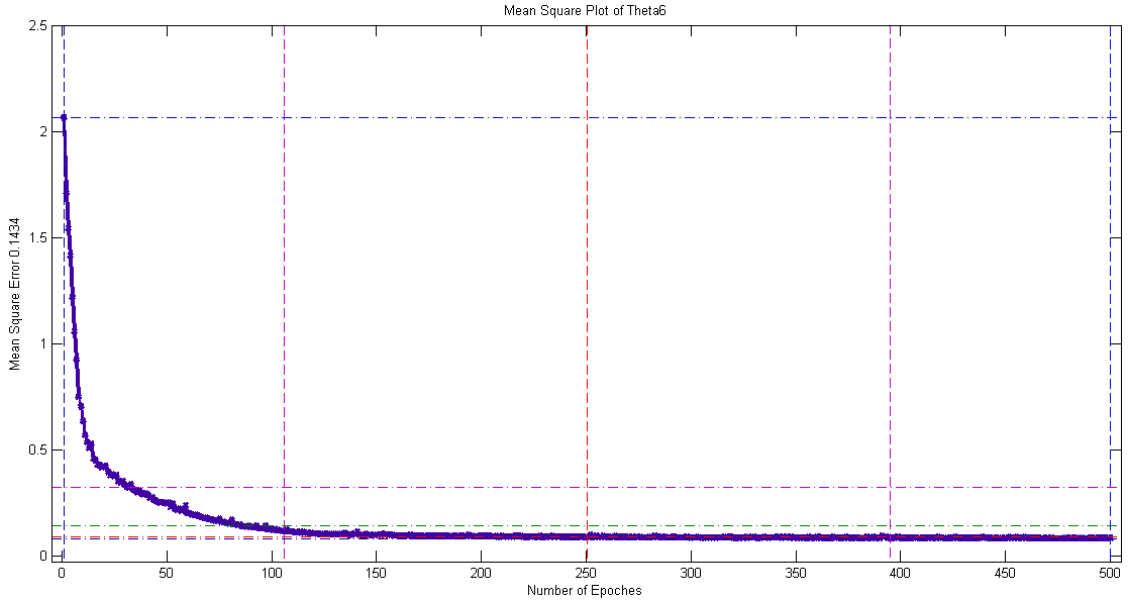
(c)


Mean Square Plot of Theta4
Mean Square Error 1.704e-2
Number of Epoches

(d)

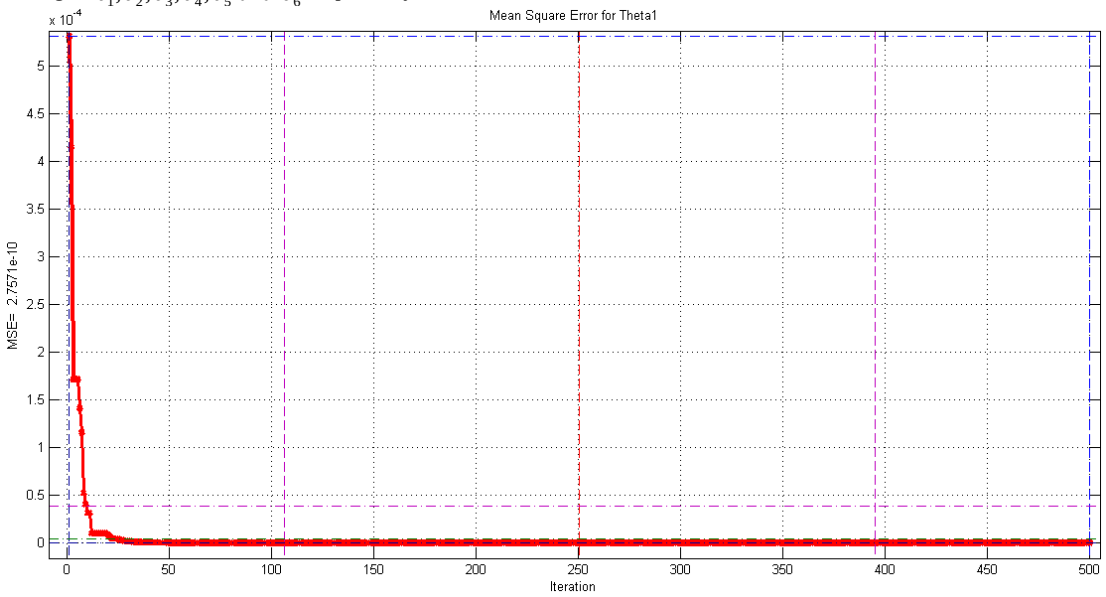


(e)

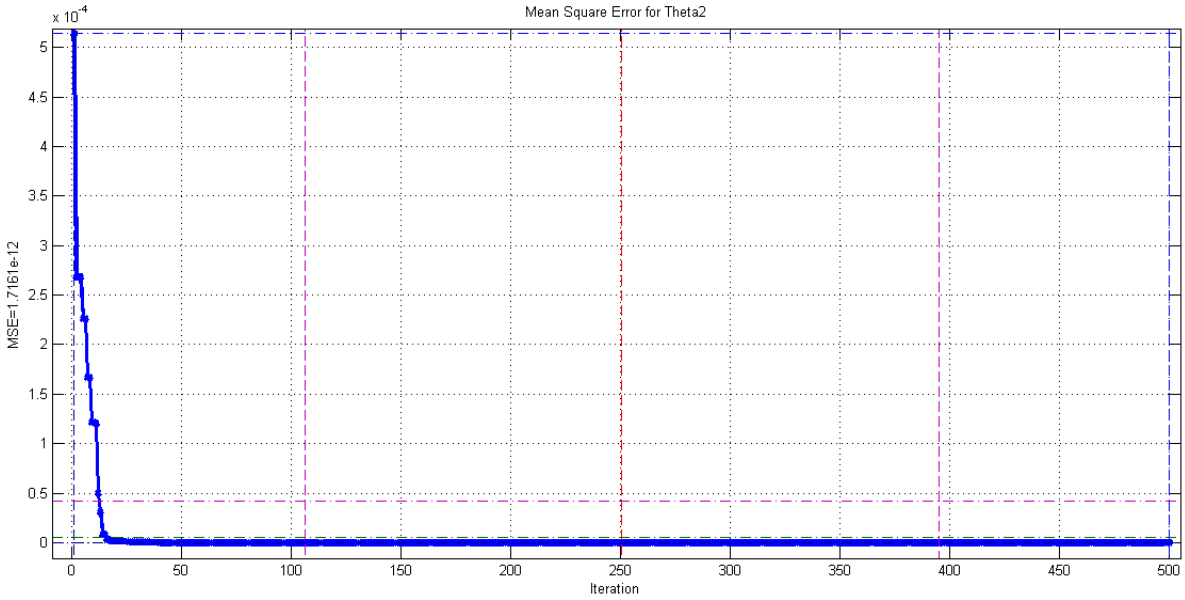


(f)

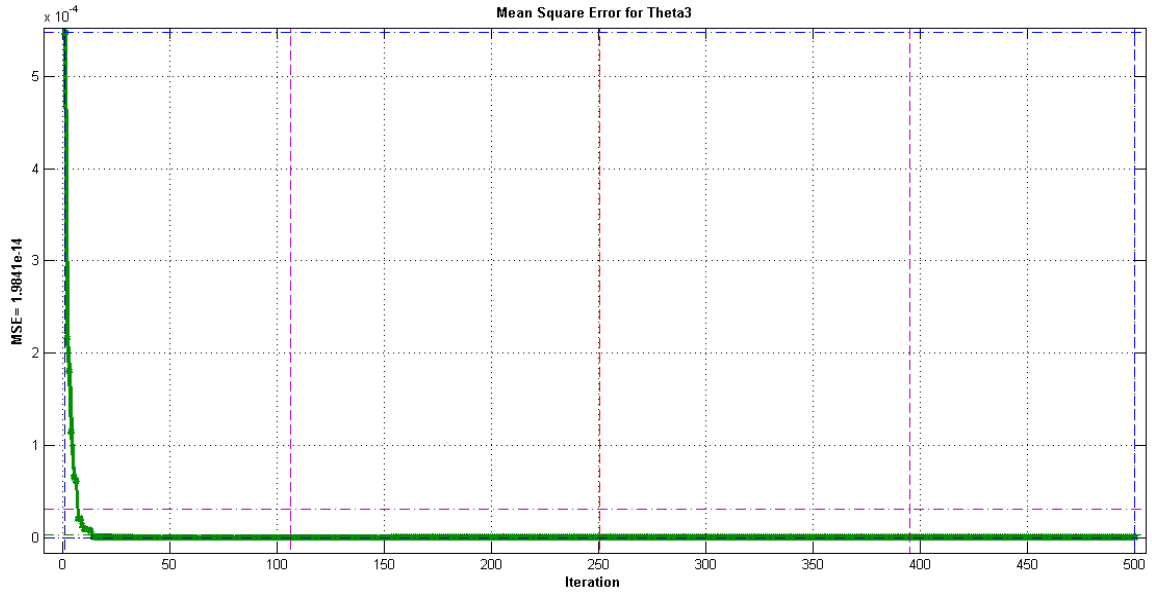
Figure 3. Mean square error for all training samples of MLPBP. Figures (a), (b), (c), (d), (e) and (f) are mean square error curve for angles $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ *and* θ_6 respectively.



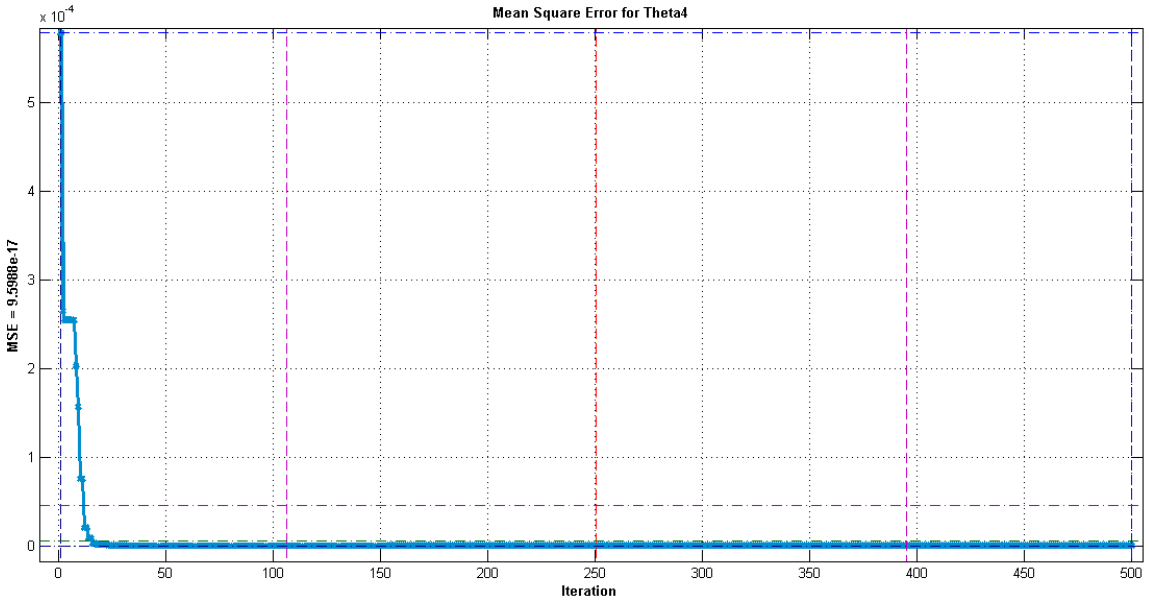
(a)


Mean Square Error for Theta2
x 10^-4
MSE=1.7161e-12
Iteration

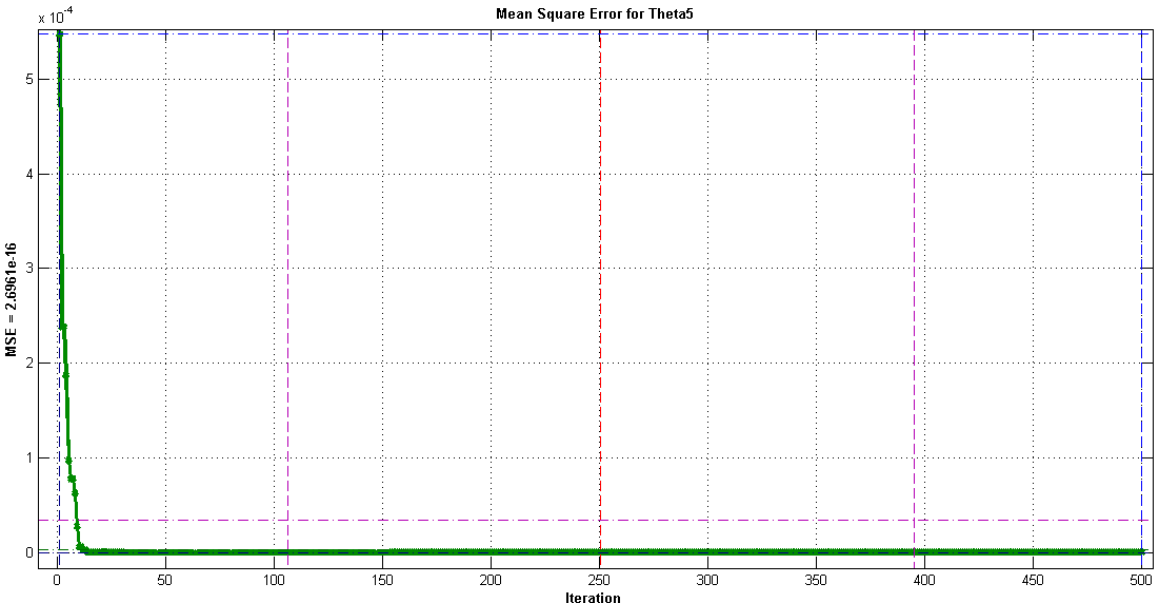
(b)


Mean Square Error for Theta3
x 10^-4
MSE= 1.9841e-14
Iteration

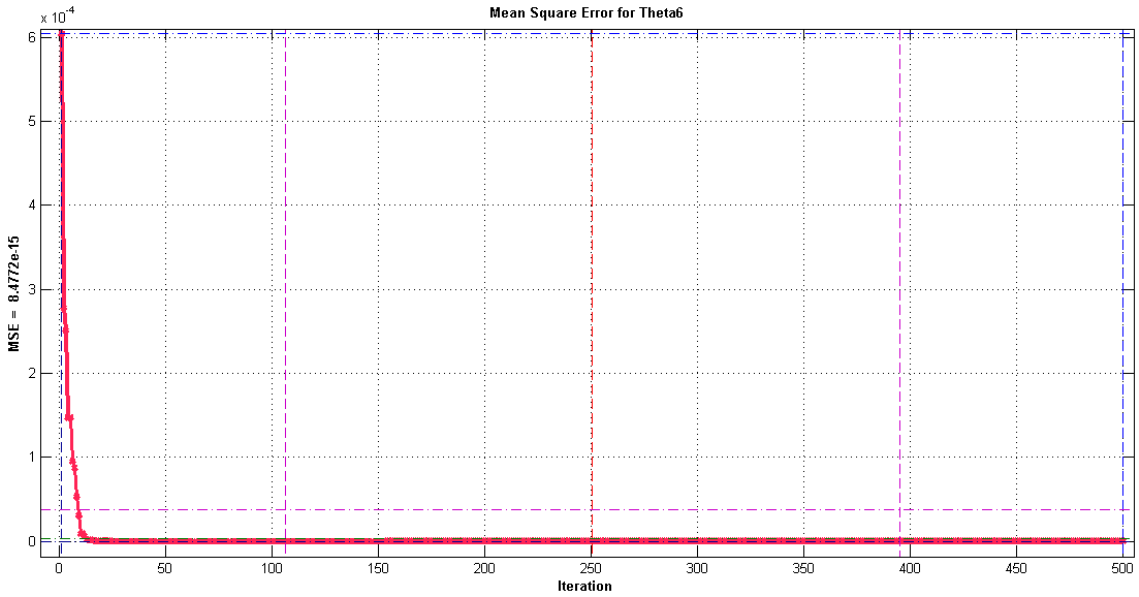
(c)


Mean Square Error for Theta4
x 10^-4
MSE = 9.5988e-17
Iteration

(d)



(e)



(f)

Figure 4. Mean square error for all training samples of MLPGSA. Figure (a), (b), (c), (d), (e) and (f) are mean square error curve for angles $\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ *and* θ_6 respectively.

V. CONCLUSION

In this paper, two methods have been considered which are MLPBP and MLPGSA 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 MLPGSA because of the fact that MLPGSA accumulates small number of epoch compared to that done by MLPBP with hybrid learning algorithm and hence MLPGSA provides a better solution for inverse kinematic problems. It has also been observed that it gives faster convergence rate and improves the problem of trapping in local minima. Therefore, MLPGSA 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 MLPGSA model.

REFERENCES

[1] D. Xu, C. A. A. Calderon, J. Q. Gan amd H. Hu, "An Analysis of the Inverse Kinematics for a 5-DOF Manipulator", *International Journal of Automation and Computing* vol. 2, pp. 114-124, June 2005.

[2] S. S Chiddarwar and N. R Babu, "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 pp. 1083–1092, March 2010.

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

[4] M. L. Husty, M. Pfurner and H. P. Schrocker, "A New and Efficient Algorithm for the Inverse Kinematics of a General Serial 6R Manipulator", *Mechanism and Machine Theory* vol. 42, pp. 66-81, February 2006.
[5] A. Olaru and S. Olaru, "Optimization of the Robots Inverse Kinematics Results by using the Neural Network and LabView Simulation", *IPCSIT*, 2011, pp. 40-45.
[6] S. Mirjalili, S. Z. M Hashim and H. M. Sardroudi, "Training Feedforward Neural Networks using Hybrid Particle Swarm Optimization and Gravitational Search Algorithm", *Applied Mathematics and Computation* vol. 218, pp. 11125–11137, July 2012.
[7] E. Rashedi, H. Nezamabadi-pour and S. Saryazdi, "GSA: A Gravitational Search Algorithm", *Information Sciences* vol. 179, pp. 2232–2248, June 2009.
[8] J. R. Zhang, J. Zhang, T. M. Lok and M. R. Lyu, "A Hybrid Particle Swarm Optimization–Back-Propagation Algorithm for Feed Forward Neural Network Training", *Applied Mathematics and Computation* vol. 185, pp. 1026–1037, February 2007.
[9] S. Sarafrazi, H. Nezamabadi-pour and S. Saryazdi, "Disruption: A New Operator in Gravitational Search Algorithm", *Scientia Iranica D* vol. 18 (3), pp. 539–548, June 2011.
[10] P.D. Wasserman, *Neural Computing*, Van Nostrand Reinhold, NewYork, 1989, pp. 110-120.

Panchanand Jha graduated in Production Engineering in the year 2007 from ITGGU, Bilaspur India. He completed Masters in Mechanical Engineering with Specialization in Production Engineering in 2009 from National Institute of Technology, Rourkela, India. After a short stint as a Lecturer in Mechanical Engineering at RCET, Bhilai he joined Department of Industrial Design, National Institute of Technology, Rourkela as a Research Fellow. His research interests include Robotics, Manufacturing Processes, soft computing techniques and Development of Optimization tools.

Bibhuti B. Biswal graduated in Mechanical Enginnering from UCE, Burla, India in 1985. Subsequently he completed his M.Tech. and Ph.D. from Jadavpur University, Kolkata. He was in faculty of Mechanical Engineering at UCE Burla from 1986 till 2004 and then joined National Institute of Technology, Rourkela as Professor and currently he is the Professor and Head of Department of Industrial Design. He has been actively involved in various research projects and published more than 90 papers at National and International levels, the areas of research being robotics, automation, maintenance engineering and industrial organization. He was a visiting Professor at MSTU, Moscow and a visiting scientist at GIST, South Korea.

Om Prakash Sahu received the B.E. degree in Electronics and Telecommunication in 2005 and M-Tech. in Instrumentation and control system in 2008 from the University of CSVT, Bhilai, India and joined as a faculty in the same institute in 2006. Currently he is pursuing PhD at National institute of technology Rourkela, India. He has been published more than 13 technical research papers at National and International levels. His areas of research interest include industrial robotics, control system and Instrumentation, Computer integrated manufacturing and automation.