International Journal of Robotics and Automation (IJRA)
Vol. 3, No. 1, March 2014, pp. 52~61
ISSN: 2089-4856

A Neural Network Approach for Inverse Kinematic of a SCARA Manipulator

Panchanand Jha, BB Biswal
Departement of Industrial Design, National Institute of Technology Rourkela, 769008, India

Article Info

Article history:

Received May 15, 2013
Revised Jan 2, 2014
Accepted Jan 25, 2014

Keyword:

D-H Parameters
Forward Kinematics
Inverse Kinematics
MLP Neural Network
SCARA Manipulator

ABSTRACT

Inverse kinematic is one of the most interesting problems of industrial robot. The inverse kinematics problem in robotics is about the determination of joint angles for a desired Cartesian position of the end effector.It comprises of the computation need to find the joint angles for a given Cartesian position and orientation of the end effectors to control a robot arm. There is no unique solution for the inverse kinematics thus necessitating application of appropriate predictive models from the soft computing domain. Artificial neural network is one such technique which can be gainfully used to yield the acceptable results. This paper proposes a structured artificial neural network (ANN) model to find the inverse kinematics solution of a 4-dof SCARA manipulator. The ANN model used is a multi-layered perceptron neural network (MLPNN), wherein gradient descent type of learning rules is applied. An attempt has been made to find the best ANN configuration for the problem. It is found that multi-layered perceptron neural network gives minimum mean square error.

Corresponding Author:

Panchanand Jha,
Departement of Industrial Design ,
National Institute of Technology, Rourkela
769008, Odisha, India.
Email: jha_ip007@hotmail.com

1. INTRODUCTION

The robot manipulator is composed of a serial chain of rigid links connected to each other by revolute and/or prismatic joints. Each robot joint location is usually defined relative to the neighbouring joint. The relation between successive joints is containing a 4x4 homogeneous transformation matrix that has orientation and position data of robots. Robot control actions are executed in the joint coordinates while robot motions are specified in the Cartesian coordinates. 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 control. In most robotic applications the desired positions and orientations of the end effectors are specified by the user in Cartesian coordinates. 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 the joint variable is denoted by i = (t), i = 1, 2, 3 .....n and position variables by xj = x(t), j = 1, 2, 3 .....m. The relations between the end-effectors position x(t) and joint angle (t)can be represented by 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 [1]. 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[2], [3].

The forward kinematic equations always have a unique solution, and the resulting Neural net can be used as a starting point for further refinement when the manipulator does become available. Artificial neural network especially MLP (multi-layered perceptron) is used to learn the forward and the inverse kinematics equations of five degrees freedom (DOF) robot arm [4]. This unsupervised method learns the functional relationship between input (Cartesian) space and output (joint) space based on a localized adaptation of the mapping, by using the manipulator itself under joint control and adapting the solution based on a comparison between the resulting locations of the manipulator's end effectors in Cartesian space with the desired location [5].

The simulation and computation of inverse kinematics using multilayer perceptron neural network is particularly useful where less computation times are needed, such as in real-time adaptive robot control [6]. If the number of degrees of freedom increases, traditional methods will become more complex and quite difficult to solve inverse kinematics [7].Many research contributions have been made related to the neural network-based inverse kinematics solution of robot manipulators [8]. The present work proposes inverse kinematics solutions based on structured MLP that can be trained quickly.

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 multi layered neural network (MLPNN) has not been tried to solve inverse kinematics problem with 4-DOF manipulator. MLP neural network is used to find inverse kinematics solution which yields multiple and precise solutions with an acceptable error and are suitable for real-time adaptive control of robotic manipulators [9]. The study of previous work shows that the most of the researchers [10] and [11]have adopted methods like ANN, ANFIS etc. for simple problem. The features of MLPNN are found quite matching and hence suitable for the present problem having complexity and involving multiple parameters. 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. The result of each network is evaluated by using direct kinematics equations to obtain information about their error. In other words, the angles obtained for each joint are used to compute the Cartesian coordinate for end effector. 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. KINEMATIC MODELING OF SCARA MANIPULATOR

The Denavit-Hartenberg (D-H) notation and methodology are used in this section to derive the kinematics of robot manipulator. The coordinate frame assignment and the DH parameters are depicted in Figure 1, and listed in Table 1 respectively, where to represents the local coordinate frames at the five joints respectively, represents the local coordinate frame at the end-effector, where θi represents rotation about the Z-axis, αi rotation about the X-axis, transition along the Z-axis, and transition along the X-axis.

Table 1. The D-H Parameters

Sl.	θ_i (degree)	d_i (mm)	a_i (mm)	α_i (degree)
1	θ_1=±120	0	a_1=250	0
2	θ_2=±130	0	a_2=150	180
3	0	d_3=150	0	0
4	θ_4	d_4=150	0	0

The transformation matrix Ai between two neighbouring frames Oi−1 and Oi is expressed in equation (1) as,

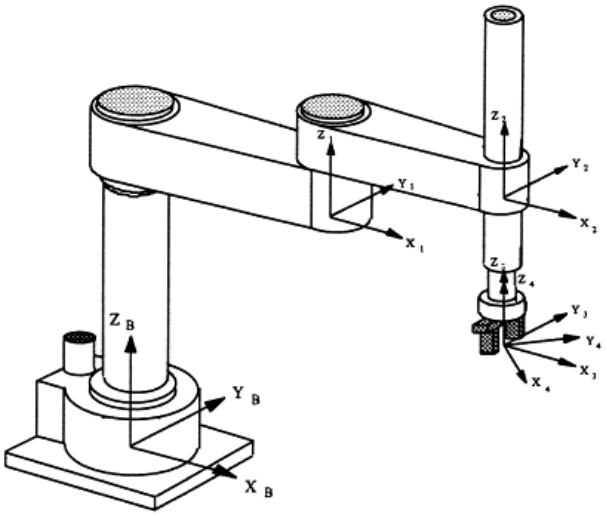

Figure 1. D-H frames of the SCARA robot.

$$A_i=\begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

By substituting the D-H parameters in Table 1 into equation (3), the individual transformation matrices A_1 to A_4 can be obtained and the general transformation matrix from the first joint to the last joint of the manipulator can be derived by multiplying all the individual transformation matrices($^0T_4$).

$$^0T_4 = A_1A_2A_3A_4 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

Where (p_x, p_y, p_z) represents the position and $\{(n_x, n_y, n_z), (o_x, o_y, o_z), \text{ and } (a_x, a_y, a_z)\}$ represents the orientation of the end-effector. The orientation and position of the end-effector can be calculated in terms of joint angles and the D-H parameters of the manipulator are shown in following matrix as:

$$\begin{bmatrix} c_{124} & -s_{124} & 0 & a_1c_1 + a_2c_{12} \\ s_{124} & c_{124} & 0 & a_1s_1 + a_2s_{12} \\ 0 & 0 & 1 & -d_3 - d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$\theta_2 = \tan^{-1}\left(\frac{s_2}{c_2}\right) = \tan^{-1}\left[\frac{\pm 2a_1a_2\sqrt{1-c_2^{\,2}}}{p_x^{\,2} + p_y^{\,2} - a_1^{\,2} - a_2^{\,2}}\right] \quad (6)$$

$$\theta_1 = \tan^{-1}\left(\frac{s_1}{c_1}\right) = \tan^{-1}\left[\frac{(a_1 + a_2c_2)p_y - a_2s_2p_x}{(a_1 + a_2c_2)p_x + a_2s_2p_y}\right] \quad (7)$$

$$d_3 = -p_z - d_4 \quad (8)$$

$$\theta_4 = \tan^{-1}\left[\frac{-n_xs_{12} + n_yc_{12}}{n_xc_{12} + n_ys_{12}}\right] \quad (9)$$

It is obvious from the representation given in equations (7) through (9) that there exist multiple solutions to the inverse kinematics problem. The above derivations with various conditions being taken into account provide a complete analytical solution to inverse kinematics of arm. So to know which solution holds good to study the inverse kinematics, all joints variables are obtained and compared using forward kinematics solution. This process is been applied for θ_1, θ_2, d_3 *and* θ_4 , to choose the correct solution, all the four sets of possible solutions (joint angles) calculated, which generate four possible corresponding positions and orientations using the forward kinematics.

3. MULTI-LAYERED PERCEPTRON NEURAL NETWORK APPLICATION FOR SCARA MANIPULATOR

It is well known that neural networks have the better ability than other techniques to solve various complex problems. Inverse kinematics is a transformation of a world coordinate frame (X, Y, and Z) to a link coordinate frame ($\theta_1, \theta_2, d_3,$ *and* θ_4). This transformation can be performed on input/output work that uses an unknown transfer function. MLP neural network's neuron is a simple work element, and has a local memory. A neuron takes a multi-dimensional input, and then delivers it to the other neurons according to their weights. This gives a scalar result at the output of a neuron. The transfer function of an MLP, acting on the local memory, uses a learning rule to produce a relationship between the input and output. For the activation input, a time function is needed.

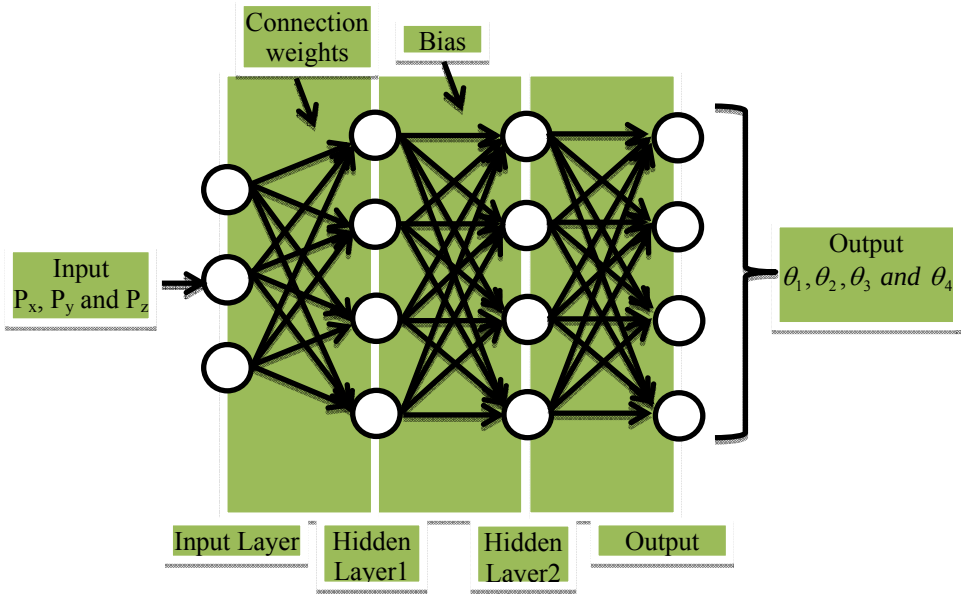

Figure 2. Multi-layered perceptron neural network structure

We propose the solution using a multi-layered perceptron with back-propagation algorithm 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.

A block diagram of the structure is shown in Figure 2. The signals, O_{jn}, are presented to a hidden layer neuron in the network via the input neurons. Each of the signals from the input neurons is multiplied by the value of the weights of the connection, w_j, between the respective input neurons and the hidden neuron.

The network uses a learning mode, in which an input is presented to the network along with the desired output and the weights are adjusted so that the network attempts to produce the desired output. Weights after training contain meaningful information whereas before training they are random and have no meaning.

Net input of hidden neurons (for *k* inputs) =

$$n_h = \sum_{j=1}^{k} w_j \times o_{jn} \tag{10}$$

The output, $O_{mj}$of a hidden neuron as a function of its net input is described in equation (10). The sigmoid function is:

$$Output = o_{mj} = \frac{1}{1+e^{-n_h}} \tag{11}$$

Once the outputs of the hidden layer neurons have been calculated, the net input to each output layer is calculated in a similar manner as in equation (11).

$$\delta = f'(n)(d - o_m) \tag{12}$$

$$\delta = o_m(1-o_m)(d-o_m) \tag{13}$$

Where d is the target or desired value, and $O_m$is the actual value from output neuron after going through the feed forward calculation. The error calculation was implemented on a neuron-by-neuron basis over the entire set (epoch) of patterns. This error value δ was used to perform the appropriate weight adjustments of the weight connection between the output layer and hidden layer.

$$\delta_h = f'(n_h)\sum_{l=0}^{k_l} w_{lh}\delta_l = o_h(1-o_h)\sum_{l=0}^{k_l} w_{lh}\delta_l \tag{14}$$

Where $\delta_h$the error value of the hidden layer is, $\delta_l$is the error value of the output layer, $O_h$is the output of the sigmoid function and $W_{lh}$is the connection weights between the output and hidden layers. The weight changes were calculated according to equation (14).

$$w(old) = w(new) + \eta\delta_0 + \alpha[w(old)] \tag{15}$$

The aim of the training phase is to minimize this average sum squared error over all training patterns. The speed of convergence of the network depends on the training rate, η and the momentum factor, α. In this work, a two hidden layer neural network with three inputs,P_x,P_y and P_z, and four outputs, θ_1, θ_2, d_3, and θ_4 was trained using the back-propagation algorithm described earlier, along a trajectory of the end-effector in the x-y plane.

4. RESULTS AND PERFORMANCE ANALYSIS

The proposed work is performed on the Matlab Neural Networks Toolbox. In this work the training data sets were generated by using equations (3) through (9). A set of 1000 data was first generated as per the formula for the input parameter P_x, P_y and P_z coordinates in mm. These data sets were the basis for the training, evaluation and testing the MLP model. Out of the sets of 1000 data, 900 were used as training data and 100 were used for testing for MLP as shown in Table 2. The following parameters were taken:

Table 2. Configuration of MLPNN

Sl.	Parameters	Values taken
1	Learning rate	0.143
2	Momentum parameter	0.43
3	Number of epochs	10000
4	Number of hidden layers	2
5	Number of inputs	3
6	Number of output	4
7	Target datasets	1000
8	Testing datasets	900
9	Training datasets	100

Back-propagation algorithm was used for training the network and for updating the desired weights. In this work epoch based training method was applied.The formulation of the MLPNN model is a generalized one and it can be used for the solution of forward and inverse kinematics problem of manipulator of any configuration. However, a specific configuration has been considered in the present work only to illustrate the applicability of the method and the quality of the solution vis-à-vis other alternatives methods.

Table 3 gives the data for position of joints determined through analytical solution and that obtained from MLPNN model.

Table 3. Comparison between analytical solution and MLPNN solution

S.N.	position of joints determined through analytical method				position of joints determined through MLPNN method			
	θ_1	θ_2	d_3	θ_4	θ_1	θ_2	d_3	θ_4
1	-120	-130	0	0	-113.1197	-120.4097	1.9361	12.3654
2	-119.7598	-129.7397	0.1502	0.3604	-113.0111	-120.3419	1.9116	11.1141
3	-119.5195	-129.4795	0.3003	0.7207	-112.898	-120.2708	1.918	9.1894
4	-119.2793	-129.2192	0.4505	1.0811	-112.7791	-120.1963	1.9657	7.2874
5	-119.039	-128.959	0.6006	1.4414	-112.6539	-120.1186	2.0591	6.3943
6	-118.7988	-128.6987	0.7508	1.8018	-112.5221	-120.038	2.1975	6.7014
7	-118.5586	-128.4384	0.9009	2.1622	-112.3838	-119.955	2.3748	7.4709
8	-118.3183	-128.1782	1.0511	2.5225	-112.2392	-119.8702	2.5785	8.1229
9	-118.0781	-127.9179	1.2012	2.8829	-112.0893	-119.7842	2.7859	8.5513
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.
10	-117.8378	-127.6577	1.3514	3.2432	-111.9351	-119.6977	2.9617	8.8311
11	-117.5976	-127.3974	1.5015	3.6036	-111.7779	-119.6114	3.0593	9.0413
12	-117.3574	-127.1371	1.6517	3.964	-111.6191	-119.5259	3.0365	9.2422
13	-117.1171	-126.8769	1.8018	4.3243	-111.4604	-119.4418	2.8917	9.4813
14	-116.8769	-126.6166	1.952	4.6847	-111.303	-119.3594	2.6945	9.7883
15	-116.6366	-126.3564	2.1021	5.045	-111.1486	-119.2791	2.5521	10.1545
16	-116.3964	-126.0961	2.2523	5.4054	-110.998	-119.2007	2.5246	10.5178
17	-116.1562	-125.8358	2.4024	5.7658	-110.8523	-119.1243	2.6043	10.7578
18	-115.9159	-125.5756	2.5526	6.1261	-110.7119	-119.0495	2.7826	10.7483
19	-115.6757	-125.3153	2.7027	6.4865	-110.577	-118.9759	3.0739	10.5484
20	-115.4354	-125.0551	2.8529	6.8468	-110.4473	-118.9028	3.454	10.6648

Table 4. Regression analysis

Sl.	Regression Coefficient (R)	Mean Square Error	Epoch Number	Resolution through AdeptOne robot with smart controller user's guide	Resolution through MLPNN
1	0.99824	0.0076	2632	0.00078^0	0.000778^0
2	0.99519	0.00471	10000	0.00312^0	0.003104^0
3	0.99972	0.00028	10000	0.0033mm	0.003299mm
4	0.99928	0.00072	10000	0.047^0	0.046966^0

This is consistent with our claim that it is a good strategy to train the ANN with a good representative set of fixed targets positions instead of variable target positions for the learning process that will introduce noise in the cost function and may result in poor convergence.

The mean square curves shown in Figure 3 through Figure 6exhibit the building knowledge procedure for the new path that gives an indication for the success of the proposed model. As shown in result, the used solution method gives the chance of selecting the output, which has the least error in the system. Hence, the solution can be obtained with less error as shown in Figures (3) through (6) for the best validation performance of the obtained data with the desired data.

Generalization tests were carried out with random target positions showing that the learned MLP generalize well over the whole space. From Table 4 we can understand the mean square error for all joint variables is quite closer to zero. The regression coeffiecint analysis as per Table 4that shows 99.9% matching for all joint variables which is acceptable for obtaiting inverse kinematics of the SCARA manipulator.Resolutions of the AdeptOne SCARA robot given in Table 4 (obtained from AdeptOne robot with smart controller user's guide) are compared with the resolution obtained from the MLPNN model. Figure 7 represents the graphical view of regression analysis.

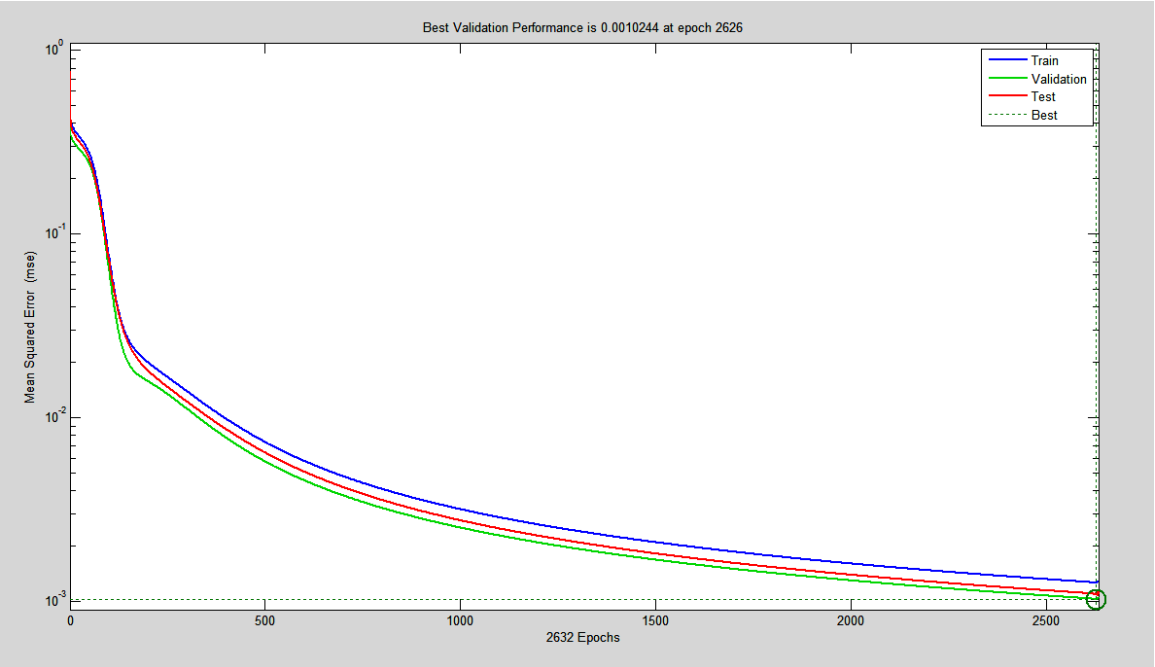

Figure 3. Mean square error for θ_1

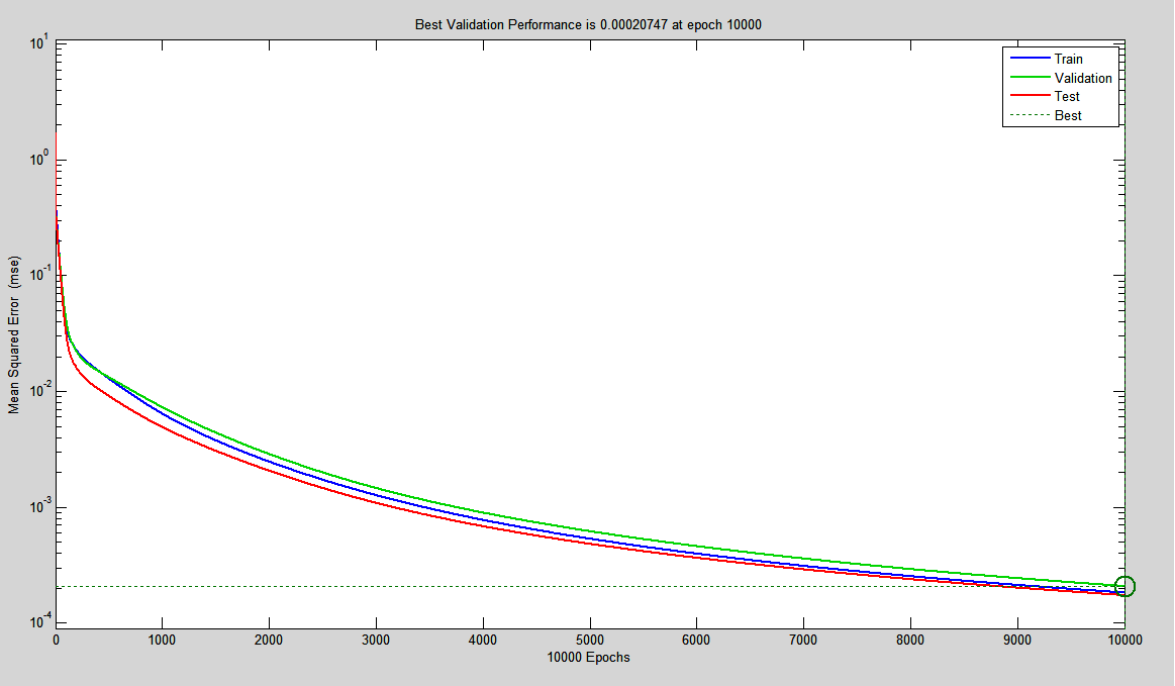

Figure 4. Mean square error for θ_2

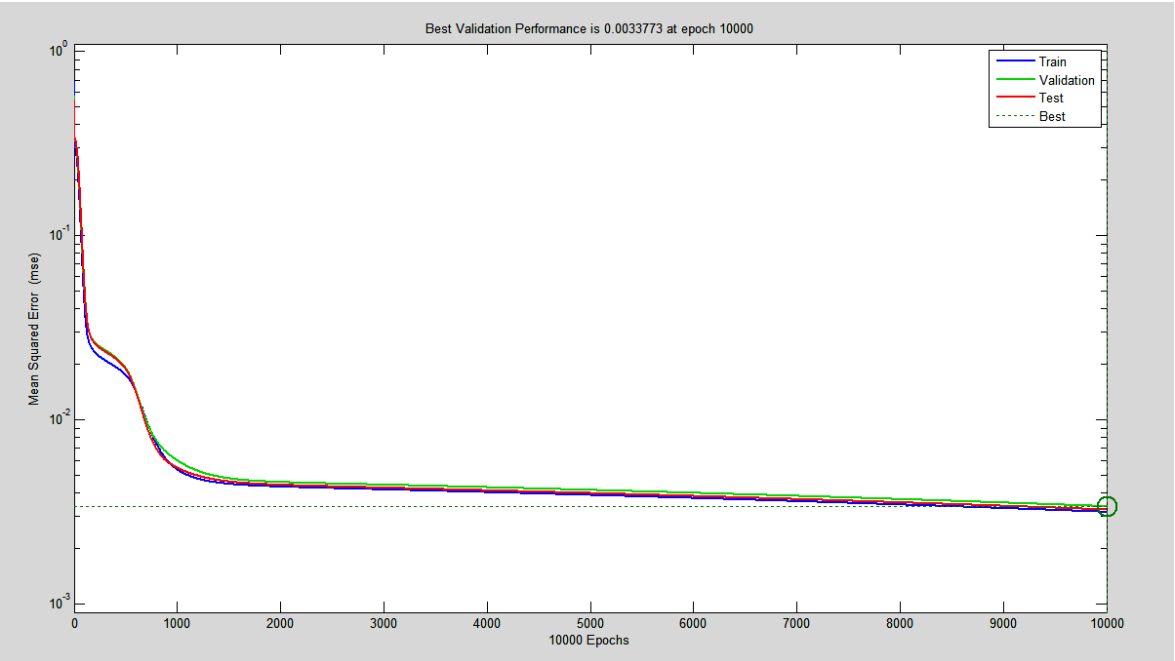

Figure 5. Mean square error for d_3

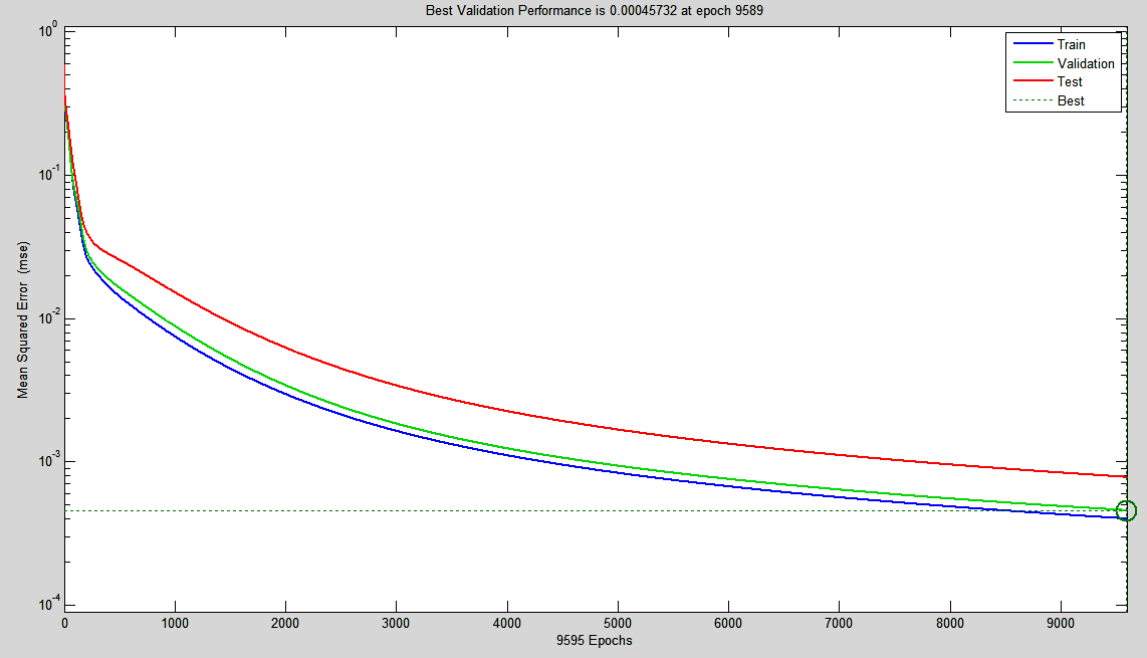

Figure 6. Mean square error for θ_4

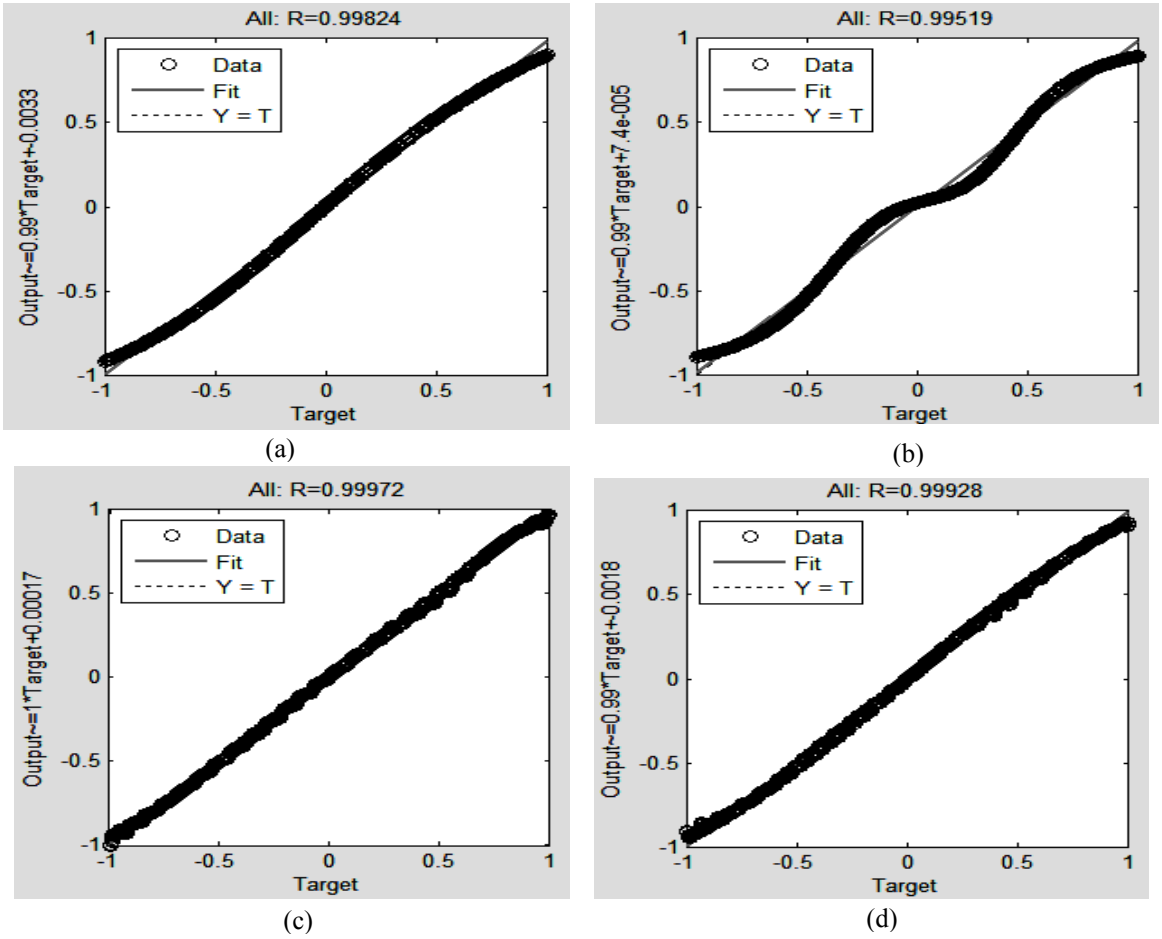

Figure 7. Graphical view of regression

5. CONCLUSION

The emphasis of this paper was mainly on the applicationof analytical and neural network solution of inverse kinematics of 4-dof SCARA robot manipulator. Mathematical models rely on assuming the structure of the model in advanced, which may be sub-optimal. Consequently many mathematical models fail to simulate the complex behaviour of inverse kinematics problem. In contrast, ANN is based on the input/output data pairs to determine the structure and parameters of the model. Moreover, they can always be updated to obtain better results by presenting new training examples as new data become available. In the present problem the error value (mean square error) is nearly zero which is very much acceptable when compare to the precision figures and repeatability error values of any typical manipulator. From the present study, it is observed that the MLP gives minimum mean square error for resolution and joints variables as performance index. This artificial neural network based joint angles prediction model can be a useful tool for the production engineers to estimate the motion of the manipulator accurately.

REFERENCES

[1] MS Alshamasin, et al. "Modelling and simulation of a SCARA robotusing solid dynamics and verification byMATLAB/Simulink". *International Journal of Modelling, Identification and Control*. 2012; 15(1).
[2] MA Al-Khedher, et al. "SCARA Robot Control using Neural Networks". *2012 4th International Conference on Intelligent and Advanced Systems* (ICIAS 2012).
[3] SM Raafat*, et al.* "Improving Trajectory Tracking of aThree Axis SCARA Robot UsingNeural Networks". *2009 IEEE Symposium on Industrial Electronics and Applications (ISIEA 2009),* October 4-6, 2009, Kuala Lumpur, Malaysia
[4] R Koker. "Reliability-based approach to the inverse kinematics solution of robots using Elman's networks", *Engineering Applications of Artificial Intelligence* 18, 685–693, 2005.
[5] A Srinivasan and MJ Nigam. "Neuro-Fuzzy based Approach for Inverse Kinematics Solution of Industrial Robot Manipulators". *Int. J. of Computers, Communications & Control*. 2008; 3: 224-234.
[6] AT Hasan, et al., "An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator". *Advances in Engineering Software*. 2006; 37: 432-438.
[7] ML Husty, M Pfurner and HP Schrocker. "A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator". *Mechanism and Machine Theory*. 2007; 42: 66-81.
[8] AT Hasan, et al. "Artificial neural network-based kinematics Jacobian solution for serial manipulator is passing through singular configurations". *Advances in Engineering Software*. 2010; 41: 359-367.
[9] A Olaru, et al. "*Assisted Research and Optimization of the proper Neural Network Solving the Inverse Kinematics Problem*". Proceedings of 2011 International Conference on Optimization of the Robots and Manipulators.
[10] WM Jasim. "Solution of Inverse Kinematics for SCARAManipulator Using Adaptive Neuro-Fuzzy Network". *International Journal on Soft Computing (IJSC)*. 2011; 2(4).
[11] RV Mayorga and P Sanongboon. "Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: An Artificial Neural Network approach". *Robotics and Autonomous Systems*. 2005; 53; 164-176.

BIOGRAPHIES OF AUTHORS

Panchanand Jha graduated in Production Engineering in the year 2007 from ITGGU, Bilaspur India. He has 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.

Dr. BB 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.