Inverse Kinematic Solution of 5R Manipulator Using ANN and ANFIS

Panchanand Jha, B. B. Biswal
Department of Industrial Design, National Institute of Technology Rourkela, India

Article Info

Article history:

Received Feb 9, 2015
Revised Apr 25, 2015
Accepted May 12, 2015

Keyword:

ANFIS
D-H Parameters
Forward Kinematics
Inverse Kinematics
MLP Neural Network

ABSTRACT

Inverse kinematics of manipulator comprises the computation required to find the joint angles for a given Cartesian position and orientation of the end effector. There is no unique solution for the inverse kinematics thus necessitating application of appropriate predictive models from the soft computing domain. Artificial neural network and adaptive neural fuzzy inference system techniques can be gainfully used to yield the desired results. This paper proposes structured artificial neural network (ANN) model and adaptive neural fuzzy inference system (ANFIS) to find the inverse kinematics solution of robot 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 ANFIS gives better result and minimum error as compared to ANN.

Corresponding Author:

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

1. INTRODUCTION

Robot manipulator is composed of a serial chain of rigid links connected to each other by revolute or prismatic joints. Each robot joint location is usually defined relative to the neighbouring joint. The relation between successive joints is described by 4x4 homogeneous transformation matrices that have orientation and position data of robots. The number of those transformation matrices determines the degrees of freedom of robots. The product of such matrices produces final orientation and position data of an n degrees of freedom robot manipulator. 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

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

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]. 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 [3-5].

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 [6-7]. 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 [6].

Many works have been completed related to the neural network-based inverse kinematics solution of robot manipulators [8-11]. The present work proposes inverse kinematics solutions based on structured MLPNN that can be trained quickly.

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 [12]. Therefore, the main aim of this work is focused on minimizing the mean square error of the neural network-based as well as ANFIS based solution of inverse kinematics problem. The result of each technique is evaluated by using inverse 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 for ANN and ANFIS have been selected very precisely. Especially, unlearned data in each neural network and ANFIS have been chosen, and used to obtain the training set of the last network.

2. KINEMATIC MODELING OF 5R MANIPULATOR

The Denavit-Hartenberg (D-H) notation and methodology are used in this section to derive the kinematics of robot manipulator. The Denavit/Hartenberg (or D-H) technique has become the standard method in robotics for describing the forward kinematics of a manipulator. Essentially, by careful placement of a series of coordinate frames fixed in each link, the D-H technique reduces the forward kinematics problem to that of combining a series of straightforward consecutive link-to-link transformations from the base to the end effector frame. Using this method, the forward kinematics for any manipulator is summarized in a table of parameters (the D-H parameters). 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 [1], [3].

Table 1. The D-H Parameters

Frame	θ_i (degree)	d_i (mm)	a_i (mm)	α_i (degree)
O_0 - O_1	θ_1	d_1= 150	a_1= 60	-90
O_1– O_2	θ_2	0	a_2= 145	0
O_2– O_3	-90 + θ_3	0	0	-90
O_3– O_4	θ_4	d_2= 125	0	90
O_4– O_5	θ_5	0	0	-90
O_5– O_6	0	d_3= 130	0	0

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

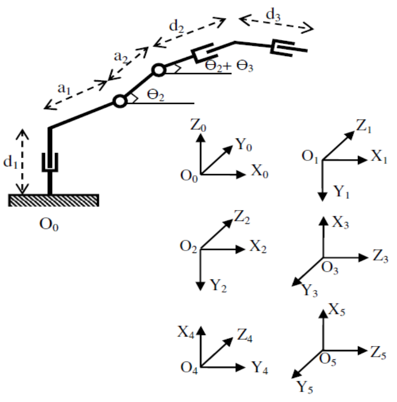


Figure 1. D-H frames of the SCARA robot.

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

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 ($^0T_4$).

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

Where (p_x, p_y, p_z) represents the position and $\{(n_x,n_y,n_z),(o_x,o_y,o_z),\ and (a_x,a_y,a_z)\}$ 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:

$$\begin{bmatrix} \begin{matrix} c_1s_{23}c_4c_5 \\ +s_1s_4c_5 \\ +c_1c_{23}s_5 \end{matrix} & \begin{matrix} -c_1s_{23}s_4 \\ +s_1c_4 \end{matrix} & \begin{matrix} -c_1s_{23}c_4s_5 \\ -s_1s_4s_5 \\ +c_1c_{23}c_5 \end{matrix} & \begin{matrix} -d_3c_1s_{23}c_4s_5 \\ -d_3s_1s_4s_5+d_3c_1c_{23}c_5 \\ +d_2c_1c_{23}+a_2c_1c_2+a_1c_1 \end{matrix} \\ \begin{matrix} s_1s_{23}c_4c_5 \\ -c_1s_4c_5 \\ +s_1c_{23}s_5 \end{matrix} & \begin{matrix} -s_1s_{23}s_4 \\ -c_1c_4 \end{matrix} & \begin{matrix} -s_1s_{23}c_4s_5 \\ +c_1s_4s_5 \\ +s_1c_{23}c_5 \end{matrix} & \begin{matrix} -d_3s_1s_{23}c_4s_5+ \\ d_3c_1s_4s_5+d_3s_1c_{23}c_5 \\ +d_2s_1c_{23}+a_2s_1c_2+a_1s_1 \end{matrix} \\ \begin{matrix} c_{23}c_4c_5 \\ -s_{23}s_5 \end{matrix} & -c_{23}s_4 & \begin{matrix} -c_{23}c_4s_5 \\ -s_{23}c_5 \end{matrix} & \begin{matrix} -d_3c_{23}c_4s_5 \\ -d_3s_{23}c_5-d_2s_{23} \\ -a_2s_2+d_1 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

where, $c_i = \cos(\theta_i),\ s_i = \sin(\theta_i),$
$c_{23} = \cos(\theta_2+\theta_3)\ and\ s_{23} = \sin(\theta_2+\theta_3)$

By equalizing the matrices in equation (4) and (5), the following equations are derived

$$p_x = -d_3c_1s_{23}c_4s_5 - d_3s_1s_4s_5 + d_3c_1c_{23}c_5 + d_2c_1c_{23} + a_2c_1c_2 + a_1c_1 \tag{6}$$

$$p_y = -d_3 s_1 s_{23} c_4 s_5 + d_3 c_1 s_4 s_5 + d_3 s_1 c_{23} c_5 + d_2 s_1 c_{23} + a_2 s_1 c_2 + a_1 s_1 \tag{7}$$

$$p_z = -d_3 c_{23} c_4 s_5 - d_3 s_{23} c_5 - d_2 s_{23} c_5 - d_2 s_{23} - a_2 s_2 + d_1 \tag{8}$$

$$n_x = c_1 s_{23} c_4 c_5 + s_1 s_4 c_5 + c_1 c_{23} s_5 \tag{9}$$

$$n_y = s_1 s_{23} c_4 c_5 - c_1 s_4 c_5 + s_1 c_{23} s_5 \tag{10}$$

$$n_z = c_{23} c_4 c_5 - s_{23} s_5 \tag{11}$$

$$o_x = -c_1 s_{23} s_4 + s_1 c_4 \tag{12}$$

$$o_y = -s_1 s_{23} s_4 - c_1 c_4 \tag{13}$$

$$o_z = -c_{23} s_4 \tag{14}$$

$$a_x = -c_1 s_{23} c_4 s_5 - s_1 s_4 c_5 + c_1 c_{23} c_5 \tag{15}$$

$$a_y = -s_1 s_{23} c_4 s_5 + c_1 s_4 s_5 + s_1 c_{23} c_5 \tag{16}$$

$$a_z = -c_{23} c_4 s_5 - s_{23} c_5 \tag{17}$$

From equations (6) through (17), the position and orientation of the arm end-effector can be calculated and provide all the joint angles. This gives solution to the forward kinematic problem. It is obvious that the inverse kinematics solution is difficult to obtain. This work uses various tricky strategies to solve the inverse kinematics of the robot manipulator.

From equations (6) and (15), the following equation is derived:

$$p_x - d_3 a_x = c_1 (d_2 c_{23} + a_2 c_2 + a_1) \tag{18}$$

Similarly by manipulating in similar way from (7) and (16), the following equation is derived as:

$$p_y - d_3 a_y = s_1 (d_2 c_{23} + a_2 c_2 + a_1) \tag{19}$$

It can be noted that the values of θ_2 and θ_3 in robot manipulator only takes integral values in a limited range. By checking all possible joint angles θ_2 and θ_3 that $(d_2 c_{23} + a_2 c_2 + a_1) \neq 0$ holds good, which means that $p_x - d_3 a_x$ and $p_y - d_3 a_y$ will not equals to zero at same time. If $(d_2 c_{23} + a_2 c_2 + a_1) > 0$, the solution for θ_1 is,

$$\theta_1 = a\tan 2(p_y - d_3 a_y, p_x - d_3 a_x) \tag{20}$$

Otherwise,

$$\theta_1 = a\tan 2(d_3 a_y - p_y, d_3 a_x - p_x) \tag{21}$$

For deriving solutions for θ_2 and θ_3, (18) and (19) can be represented as follows:

$$d_2 c_{23} + a_2 c_2 = (p_x - d_3 a_x) / c_1 - a_1 \tag{22}$$

$$d_2c_{23} + a_2c_2 = (p_y - d_3a_y)/s_1 - a_1 \tag{23}$$

From (8) and (17), the following equation can be derived:

$$p_z - d_3a_z = -d_2s_{23} - a_2s_2 + d_1 \tag{24}$$

Now considering (22) and (24),
Let

$$r = (p_x - d_3a_x)/c_1 - a_1 \tag{25}$$

And

$$r_z = -d_2s_{23} - a_2s_2 + d_1 \tag{26}$$

Now squaring the equations (25) and (26) followed by adding it, equation (27) can be derived as follow:

$$d_2^2 + 2a_2d_2(c_2c_{23} + s_2s_{23}) + a_2^2 = r + r_z^2 \tag{27}$$

Solving the terms $c_2c_{23} + s_2s_{23}$ in the above equation (27), we get

$$(c_2c_{23} + s_2s_{23})\cos\theta_3 = -\cos(\theta_3 - \pi)$$
$$= \cos(-\theta_3) = -\cos(\pi - \theta_3)$$

Therefore, there are several possible solutions for θ_3 , which are as follows:

$$\theta_3 = \pm\,\mathrm{acos}\left(\frac{a^2 \;\; + r_z^2 - a_2^2 - d_2^2}{2a_2d_2}\right) \tag{28}$$

Or,

$$\theta_3 = \pm\left[\pi - a\cos\left(\frac{a_2^2 - d_2^2 - r^2 + r_z^2}{2a_2d_2}\right)\right] \tag{29}$$

Now consider the possible solutions for θ_2 . For the sake of convenience, equation (24) can be rewritten as equation (30),

$$d_2s_{23} = B_1 - a_2s_2 \tag{30}$$

where, $d_2a_z - p_z + d_1 = B_1$

Considering the equations (22) and (23), equation (31) is derived as,

$$\mathrm{d_2c_{23}} + \mathrm{a_2c_2} = \pm\sqrt{(-a_xd_3 + p_x)^2 + (-a_yd_3 + p_y)^2} \tag{31}$$

Let $B_2 = \pm\sqrt{(-a_x d_3 + p_x)^2 + (-a_y d_3 + p_y)^2}$, so equation (31) can be rewritten as,

$$d_2 c_{23} = B_2 - a_2 c_2 \tag{32}$$

Rearranging equation (28), (30) and solving for B_1, B_2 . Equation (31), (32) is derived as:

$$B_1 = (d_2c_3 + a_2)\,s_2 + (d_2s_3)\,c_2 \tag{33}$$

$$B_2 = (d_2c_3 + a_2)\,c_2 - (d_2s_3)\,s_2 \tag{34}$$

Diving both side of (33) and (34), by $\sqrt{B_1^2 + B_2^2}$, equation (35) and (36) is derived as,

$$\cos\theta * \sin\theta_2 + \sin\theta * \cos\theta_2 = \frac{B_1}{\sqrt{B_1^2 + B_2^2}} \tag{35}$$

$$\cos\theta * \sin\theta_2 - \sin\theta * \cos\theta_2 = \frac{B_2}{\sqrt{B_1^2 + B_2^2}} \tag{36}$$

where, $\cos\theta = \frac{(d_2c_3 + a_2)}{\sqrt{B_1^2 + B_2^2}}$ and $\sin\theta = \frac{(d_2s_3)}{\sqrt{B_1^2 + B_2^2}}$

The equation (34) and (35) are rewritten as,

$$\sin(\theta + \theta_2) = \frac{B_1}{\sqrt{B_1^2 + B_2^2}} \tag{37}$$

And,

$$\cos(\theta + \theta_2) = \frac{B_2}{\sqrt{B_1^2 + B_2^2}} \tag{38}$$

Therefore, $\theta + \theta_2 = a\tan 2(B_1, B_2) + 2m\pi$ and $\theta = \pm\,\mathrm{acos}\frac{(d_2c_3 + a_2)}{\sqrt{B_1^2 + B_2^2}}$,

Where m = -1, 0 or1. It is clear that θ could be in $[0,\pi]$ or $[-\pi,0]$. The range of will depend on the range of θ_3. Therefore, if $0 \le \theta_3 \le \pi$, then $s_3 > 0$ and $\sin(\theta) < 0$, thus $0 \le \theta \le \pi$. Then θ_2 can be derived as:

$$\theta_2 = \mathrm{atan2}(B_1, B_2) - a\cos\frac{(d_2c_3 + a_2)}{\sqrt{B_1^2 + B_2^2}} + 2m\pi \tag{39}$$

Otherwise, if $-\pi < \theta_3 < 0$, then $s_3 < 0$ and $\sin(\theta) < 0$, thus $-\pi < \theta < 0$. Then the next possible solution for θ_2 is as:

$$\theta_2 = \mathrm{atan2}(B_1, B_2) + a\cos\frac{(d_2c_3 + a_2)}{\sqrt{B_1^2 + B_2^2}} + 2m\pi \tag{40}$$

Now that θ_1, θ_2 and θ_3 are known, the solutions for θ_4 and θ_5 can be found by using the remaining forward kinematics equations. Considering equation (14), the value of

$$s_4 = -\frac{o_z}{c_{23}}, \tag{41}$$

when $c_{23} \neq 0$

Similarly from equation (12) and (13), the possible solution for c_4 is derived as:

$$c_4 = \frac{(o_x - c_1 s_{23} o_z / c_{23})}{s_1} \tag{42}$$

And again

$$c_4 = \frac{-(o_y - s_1 s_{23} o_z / c_{23})}{c_1} \tag{43}$$

Using equation (42) and (43) for small value of c_1, the solution for θ_4 is

$$\theta_4 = a\tan 2\left(-\frac{o_z}{c_{23}}, \frac{(o_z - c_1 s_{23} o_z / c_{23})}{s_1} \right) \tag{44}$$

Otherwise for small s_1,

$$\theta_4 = a\tan 2\left(\frac{-o_z}{c_{23}}, \frac{-(o_y - s_1 s_{23} o_z / c_{23})}{c_1} \right) \tag{45}$$

Now for solution of θ_5, considering equation (11), the value of

$$c_5 = \frac{n_z + s_{23} s_5}{c_{23} c_4} \tag{46}$$

Similarly the value of s_5 is derived by using equation (17) i.e.,

$$s_5 = -\frac{a_z + s_{23} c_5}{c_{23} c_4} \tag{47}$$

Using equation (43) in (42) and vice versa, the term c_5 and s_5 is rewritten as:

$$c_5 = \frac{n_z c_{23} c_4 - s_{23} a_z}{c_{23}^2 c_4^2 + s_{23}^2} \quad \text{And} \quad s_5 = -\frac{(a_z c_{23} c_4 + s_{23} n_z)}{c_{23}^2 c_4^2 + s_{23}^2}$$

Now using this above derivation of c_5 and s_5, θ_5 is derived as follows:

$$\theta_5 = a\tan 2\{-(a_z c_{23} c_4 + s_{23} n_z), (n_z c_{23} c_4 - s_{23} a_z)\} \tag{48}$$

It is obvious from the given equations from (3) through (48) 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. It is noted that there exist two possible solutions for $\theta_1, \theta_2, \theta_3, \theta_4$ *and* θ_5 depicted in (20) or (21), (39) or (40), (28) or (29), (44) or (45) respectively. So to know which solution holds good to study the inverse kinematics, all joints angles are obtained and compared using forward kinematics solution. This process is been applied for $\theta_1, \theta_2, \theta_3, \theta_4$ *and* θ_5. 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. 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. The solutions (21), (39), (28), (45) and (48) holds correct for obtaining the values of $\theta_1, \theta_2, \theta_3, \theta_4$ *and* θ_5 respectively.

3. ARCHITECTURES OF ANN AND ANFIS

3.1 Architecture of MLPNN

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 (Px, Py, and Pz) to a link coordinate frame ($\theta_1, \theta_2, \theta_3, \theta_4$ *and* θ_5). 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 [4], [17].

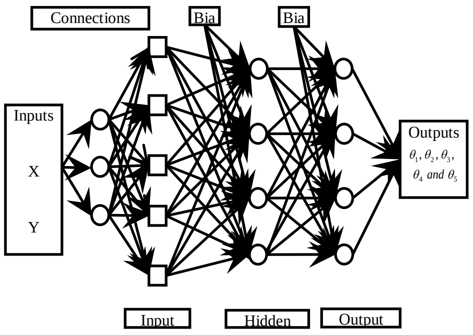


Figure 2. Multi-layered perceptron neural network structure

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) =

$$n_h = \sum_{j=1}^{k} w_j \times o_{jn} \tag{49}$$

The output, O_{mj} of a hidden neuron as a function of its net input is described in equation (49). The sigmoid function is:

$$Output = o_{mj} = \frac{1}{1+e^{-n_h}} \tag{50}$$

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 (50).

$$\delta = f^{'}(n)(d - o_m) \qquad (51)$$

$$\delta = o_m(1 - o_m)(d - o_m) \qquad (13)$$

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.

$$\delta_h = f'(n_h)\sum_{l=0}^{k_l} w_{lh}\delta_l = o_h(1 - o_h)\sum_{l=0}^{k_l} w_{lh}\delta_l \qquad (52)$$

Where δ_h the error value of the hidden layer is, δ_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 (52).

$$w(old) = w(new) + \eta\delta_0 + \alpha[w(old)] \qquad (53)$$

In this work, a five hidden layer neural network with three inputs, x, y and z, and five outputs, θ1 , θ 2, θ 3 , θ 4, and θ 5 was trained using the back-propagation algorithm described earlier, along a trajectory of the end-effector in the xy -plane.

3.2 Architecture of ANFIS

The ANFIS can perform the mapping relation between the input and output data through a learning algorithm to optimize the parameters of a given FIS. The ANFIS architecture consists of fuzzy layer, product layer, normalized layer, de-fuzzy layer, and summation layer. A typical architecture of ANFIS is shown in Fig. 3, in which a circle indicates a fixed node, whereas a square indicates an adjustable node. For example, we consider two inputs x, y and one output z in the FIS. The ANFIS used in this paper implements a first-order Sugeno FIS. Among many fuzzy systems, the Sugeno fuzzy model is the most widely applied, because of its high interpretability and computational efficiency, and built-in optimal and adaptive techniques [8].
For a first-order Sugeno fuzzy system, the typical rule set can be expressed as:

Rule 1: If x is A1 and y is B1, then z1 = p1x + q1y + r1
Rule 2: If x is A2 and y is B2, then z2 = p2x + q2y + r2

where Ai and Bi are the fuzzy sets in the antecedent, and pi, qi, and ri are the parameters that are assigned during the training procedure. As in Fig. 3, the ANFIS consists of five layers. Every ith node in the first layer is an adaptive node with a node output defined by:

$$O_i^1 = \mu_{A_i}(x), \qquad i = 1,2$$
$$O_i^1 = \mu_{B_{i-2}}(y), \quad i = 3,4$$

Where $\mu_{A_i}(x)$ and $\mu_{B_{i-2}}(y)$ can adopt any fuzzy membership function (MF). In this paper, the following Gaussian MF is used:

$$gaussmf(x, c, s) = e^{-\frac{(x-c)^2}{2s^2}}$$

Where {ci , si} is the parameter set that changes the shapes of the MF. The parameters of this layer are termed the premise parameters. Every node in the second layer is a fixed node labelled Π, whose output is the product of all the incoming inputs:

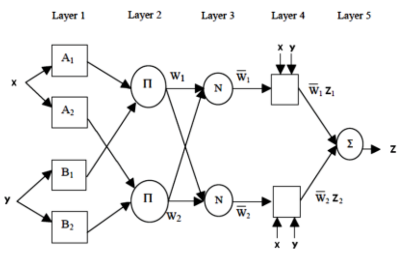


Figure 3. Architecture of ANFIS

$$O_i^2 = \omega_i = \mu_{A_i}(x)\mu_{B_i}(y), \qquad i = 1,2$$

Each node output represents the firing strength of a rule.

Every node in the third layer is a fixed node labelled N. In this layer, the average is calculated based on weights taken from fuzzy rules:

$$O_i^1 = \overline{\omega_i} = \frac{\omega_i}{\omega_1 + \omega_2}, \quad i = 1,2$$

Where ω_i is referred to as the normalized firing strengths. Every *i*th node in the fourth layer is an adaptive node with the following node function:

$$O_i^4 = \overline{\omega_i} z_i = \overline{\omega_i}(p_i x + q_i y + r_i), \qquad i = 1,2$$

Where ω_i is the output of layer 3, and {*p*i, *q*i, *ri*} is the parameter set. The parameters of this layer are termed the consequent parameters. The single node in the fifth layer is a fixed node labeled Σ that computes the overall output as the summation of all incoming inputs:

$$O_i^4 = \sum_{i=1}^{2} \overline{\omega_i} z_i = \frac{\omega_1 z_1 + \omega_2 z_2}{\omega_1 + \omega_2}$$

3.2.1 Learning algorithm

It is seen from the ANFIS architecture that when the values of the premise parameters are fixed, the output of the ANFIS can be calculated as:

$$z = \frac{\omega_1}{\omega_1 + \omega_2} z_1 + \frac{\omega_2}{\omega_1 + \omega_2} z_2$$

Substituting Eq. (5) into Eq. (8) yields:

$$z = \overline{\omega_1} z_1 + \overline{\omega_2} z_2$$

Substituting the fuzzy if-then rules into Eq. (9), it becomes:

$$z = \overline{\omega_1}(p_1 x + q_1 y + r_1) + \overline{\omega_2}(p_2 x + q_2 + r_2)$$

After rearrangement, the output can be written as a linear combination of the consequent parameters:

$$z = (\overline{\omega_1} x) p_1 + (\overline{\omega_1} y) q_1 + (\overline{\omega_1}) r_1 + (\overline{\omega_2} x) p_2 + (\overline{\omega_2} y) q_2 + (\overline{\omega_2}) r_2$$

The optimal values of the consequent parameters can be found by using the Least-Square Method (LSM). When the both premise and consequent parameters are adaptive, the search space becomes larger and the convergence of training becomes slower. The hybrid learning algorithm [12], combining the LSM and the back propagation algorithm can be used to solve this problem. This algorithm converges much faster because it reduces the dimension of the search space of the back propagation algorithm. During the learning procedure, the premise and consequent parameters are tuned until the desired response of the FIS is achieved. The hybrid learning algorithm is divided into two steps: forward pass and a backward pass. In the forward pass, while the premise parameters are held fixed, the network inputs propagated forward until layer 4, where the consequent parameters are identified by the LSM. In the backward pass, the consequent parameters are held fixed while the error signals are propagated from the output end to the input end, and the standard back propagation algorithm updates the premise parameters. Figure 3 shows the procedure of ANFIS training. This paper considers three ANFIS structure with first-order Sugeno fuzzy system for three joint angles. Figure 4 shows the structure of ANFIS model. Gaussian MFs with product inference rule are used at the fuzzification layer. Hybrid learning algorithm that combines LSM with back propagation algorithm is used to adjust the premise and consequent parameter.

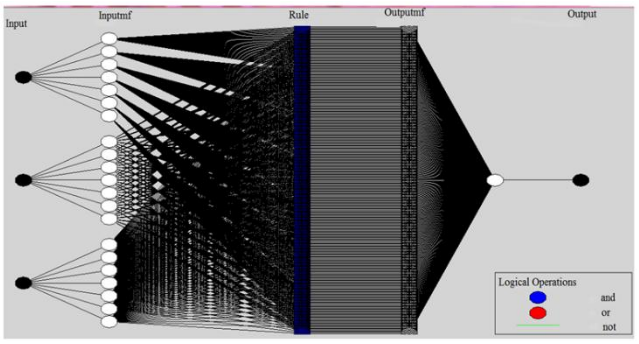


Figure 4. ANFIS model structure

4. RESULTS AND PERFORMANCE ANALYSIS

4.1 MLPNN simulation and result

The proposed work is performed on the Matlab Neural Networks Toolbox. ‘Premnmx’ function is used for preprocessing of input and outpur data. Then, the function ‘newff’ is used to create a feed forward network for inverse kinematics. Further, the same network is trained according to ‘tansig’ and ‘logsig’ transfer function. The training functions employed are ‘trainoss’ and ‘trainlm’, to validate the performance of MLPNN neural network for inverse kinematics problem. Then, the weights and biases are calculated for the network. To simulate the data corresponding to the task considered here, the new input data to the trained network are preprocessed with the ‘traimnmx’ function. Then, the outputs simulated by the trained network are post processed back using the ‘postmnmx’ function.

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 inches. These data sets were the basis for the training, evaluation and testing the MLPNN model. Out of the sets of 1000 data points, 900 were used as training data and 100 were used for testing for MLPNN as shown in table 2. The following parameters were taken:

Table 2. Configuration of MLPNN

Sl.	Parameters	Values taken
1	Learning rate	0.59
2	Momentum parameter	0.68
3	Number of epochs	10000
4	Number of hidden layers	2
5	Number of inputs	3
6	Number of output	5
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 mean square curve shown in Figure 5 through Figure 9 in result, 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. Figure 5 through Figure 9 shows the validation curve for the Problem of learning the inverse kinematics of the 5-DOF manipulator. These errors are small and the MLPNN algorithm is, therefore, acceptable for obtaining the inverse kinematics solution of the robotic manipulator. Figure 10 shows the graphical view of regression with respect to number of epochs and it’s almost gives 99.99%.

Table 4 shows comparison between the MLPNN with respect to ANFIS. Generalization tests were carried out with new random target positions showing that the learned MLPNN gives a deviation of 0.29599 of the error goal during the learning process and ANFIS gives 0.004448 average errors which is better than the mean square error of MLPNN.

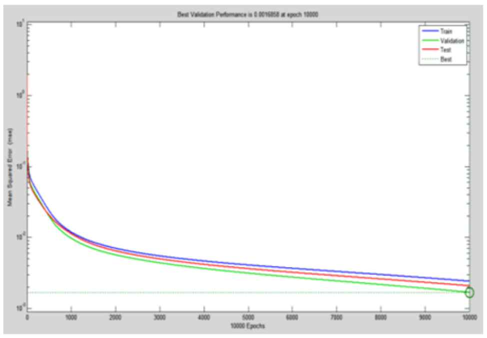


Figure 5. Mean square error for θ_1

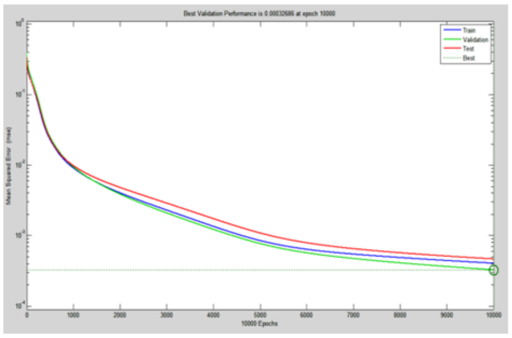


Figure 6. Mean square error for θ_2

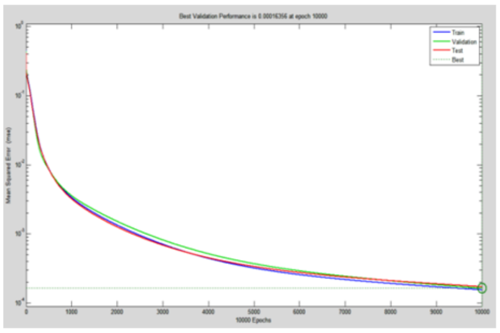


Figure 7 . Mean square error for θ_3

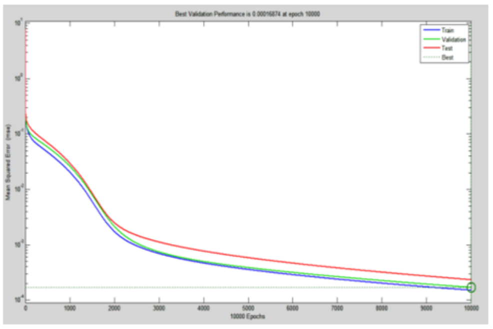


Figure 8. Mean square error for θ_4

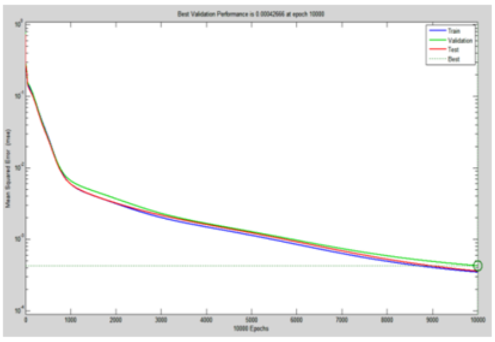


Figure 9. Mean square error for θ_5

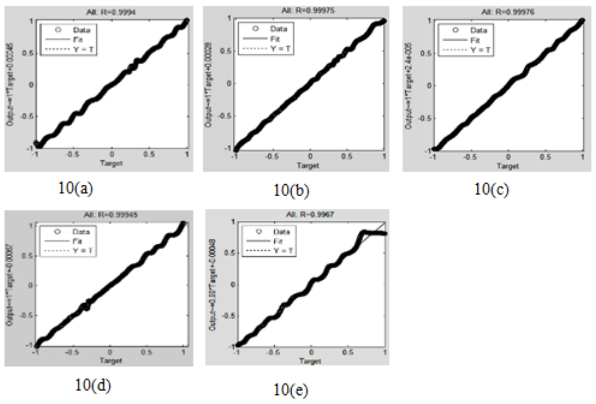


Figure 10. Regression coefficient plot for joint variables.

4.2 ANFIS simulation and results

Data set were generated by using inverse kinematic equations. The coordinates (px, py and pz) and the angles (θ1, θ2, θ3, θ4, and θ5) are used as training data to train ANFIS network with Gaussian membership function with hybrid learning algorithm. 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 ANFIS. Out of the sets of 1000 data points, 700 were used as training data and 300 were used for testing the performance of ANFIS.

In the training phase, the membership functions and the weights will be adjusted such that the required minimum error is satisfied or if the number of epochs reached. At the end of training, the trained ANFIS network would have learned the input/output map and it is tested with the deduced inverse kinematics. Figure 11 through figure 15 shows the difference in joint variables analytically and the data predicted with ANFIS.

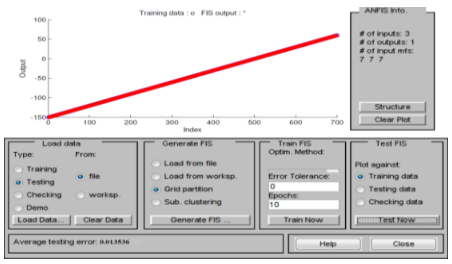
Table 3 shows configuration of ANFIS. Figure 11 through Figure 15 shows the validation curve for the problem of learning the inverse kinematics of the 5-DOF manipulator. Table 4 gives the average errors of joint variables using ANFIS. These errors are small and the ANFIS algorithm is, therefore, acceptable for obtaining the inverse kinematics solution of the robotic manipulator.

Table 3. Configuration of ANFIS

Parameters	value
Number of nodes	734
Number of linear parameters	343
Number of nonlinear parameters	63
Total number of parameters	406
Number of training data pairs	700
Number of checking data pairs	0
Number of fuzzy rules	343

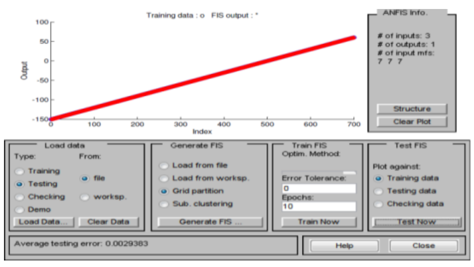
Table 4. Comparison oresults

Sl.	Average testing Error of MLPNN	Epoch Number MLPNN	Average testing Error of ANFIS	Epoch Number ANFIS
1	0.112475	10000	0.0035263	10
2	0.451253	10000	0.0029383	10
3	0.336321	10000	0.013536	10
4	0.258163	10000	0.0016652	10
5	0.321749	100000	0.00057395	10



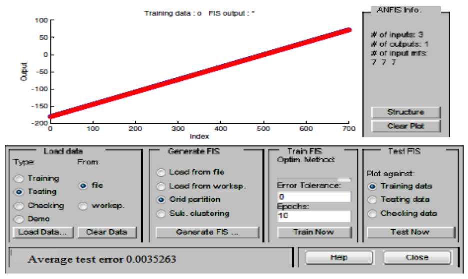
ANFIS Info.: # of inputs: 3 # of outputs: 1 # of input mfs: 7 7 7
Load data — Type: Testing; From: file
Generate FIS: Grid partition
Train FIS — Error Tolerance: 0; Epochs: 10
Test FIS — Plot against: Training data
Average testing error: 0.013536

Figure 13. Mean square error for θ_3



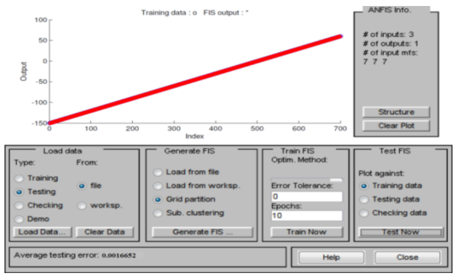
ANFIS Info.: # of inputs: 3 # of outputs: 1 # of input mfs: 7 7 7
Load data — Type: Testing; From: file
Generate FIS: Grid partition
Train FIS — Error Tolerance: 0; Epochs: 10
Test FIS — Plot against: Training data
Average testing error: 0.0029383

Figure 12. Mean square error for



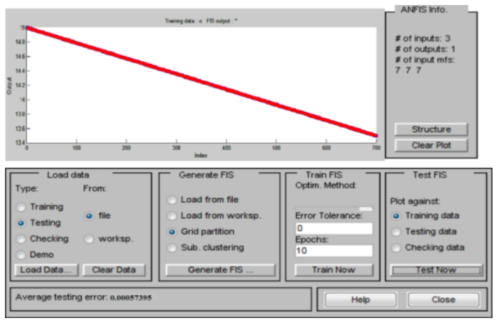
ANFIS Info.: # of inputs: 3 # of outputs: 1 # of input mfs: 7 7 7
Load data — Type: Testing; From: file
Generate FIS: Grid partition
Train FIS — Error Tolerance: 0; Epochs: 10
Test FIS — Plot against: Training data
Average test error 0.0035263

Figure 11. Mean square error for θ_1



ANFIS Info.: # of inputs: 3 # of outputs: 1 # of input mfs: 7 7 7
Load data — Type: Testing; From: file
Generate FIS: Grid partition
Train FIS — Error Tolerance: 0; Epochs: 10
Test FIS — Plot against: Training data
Average testing error: 0.0016652

Figure 14. Mean square error for



# of inputs: 3
# of outputs: 1
# of input mfs:
7 7 7

Type: Training, [✓ Testing], Checking, Demo
From: [✓ file], worksp.

Load from file, Load from worksp., [✓ Grid partition], Sub. clustering

Optim. Method:
Error Tolerance: 0
Epochs: 10

Plot against: [✓ Training data], Testing data, Checking data

Average testing error: 0.00057395

Figure 15. Mean square error for θ_5

5. CONCLUSION

In this paper, we have selected two methods which are ANN and ANFIS to obtain the solution of inverse kinematics of 5R manipulator. In this approach forward and inverse kinematic model of 5R manipulator is used to generate the data set for training the ANN and ANFIS model. The difference in desired and predicted data with ANN, gives poor results as compared to ANFIS. Also, the ANFIS accumulate small number of epoch with hybrid learning algorithm. Therefore, ANFIS 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 ANFIS model.

REFERENCES

[1] Xu D *et al.*, "An Analysis of the Inverse Kinematics for a 5-DOF Manipulator*", International Journal of Automation and Computing* vol. 2 pp. 114-124, 2005.
[2] Shital S, Chiddarwar N and Ramesh B., "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* 23 (2010) 1083–1092, 2010.
[3] Wenjun L, et al., "Numerical Study on Inverse Kinematic Analysis of 5R Serial Robot", *International Forum on Information Technology and Applications*, 2010.
[4] Jha P. and Biswal B. B., "A Neural Network Approach for Inverse Kinematic Solution of a SCARA Manipulator", *International Journal of Robotics and Automation*, Vol. 3, No. 1, pp. 52-61, 2014.
[5] Koker R et al., "Study of Neural Network Based Inverse Kinematics Solution for a Three-Joint Robot", *Robotics and Autonomous Systems* 49, 227–234, 2004.
[6] Karlik B and Aydin. "An improved approach to the solution of inverse kinematics problems for robot manipulators", *Engineering Applications of Artificial Intelligence* 13, 159-164, 2000..
[7] Koker R., "Reliability-based approach to the inverse kinematics solution of robots using Elman's networks", *Engineering Applications of Artificial Intelligence* Vol. 18, 685–693, 2005.
[8] Alavandar, Srinivasan and Nigam, M. J., "Neuro-Fuzzy based Approach for Inverse Kinematics Solution of Industrial Robot Manipulators", *Int. J. of Computers, Communications & Control*, Vol. 3, pp. 224-234, 2008..
[9] A. S. Morris and A. Mansor., "artificial neural network for finding inverse kinematics of robot manipulator using look up table", *Robotica*, vol. 15, pp. 617 – 625. 1997.
[10] Hasan A 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* Vol. 37, 432–438, 2006.
[11] Husty M L, Pfurner M and Schrocker H P., "A new and efficient algorithm for the inverse kinematics of a general serial 6R manipulator", *Mechanism and Machine Theory* Vol. 42, 66-81, 2007.
[12] Yahya S et al., "Geometrical approach of planar hyper-redundant manipulators: Inverse kinematics, path planning and workspace", *Simulation Modeling Practice and Theory*, Vol. 19, 406–422, 2011.
[13] Hasan A et al., "Artificial neural network-based kinematics Jacobian solution for serial manipulator is passing through singular configurations", *Advances in Engineering Software*, Vol. 41, 359–367, 2010.
[14] Olaru A and Olaru S., "Optimization of the robots inverse kinematics results by using the Neural Network and LabView simulation", *IPCSIT* vol.13, 2011.
[15] Mayorga R V and Sanongboon P., "Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: An Artificial Neural Network approach", *Robotics and Autonomous Systems* Vol. 53, 164–176, 2005.
[16] Xie J et al., *"Inverse Kinematics Problem for 6-DOF Space Manipulator Based on the Theory of Screws*", Proceedings of the IEEE International Conference on Robotics and Biomimetic, 2007.
[17] Haykin S., Neural networks a comprehensive foundation, 1994.

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. B. 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.