I J C T A, 9(7), 2016, pp. 3211-3229
© International Science Press ISSN: 0974-5572

Inverse Kinematics Solution of a Five Joint Robot Using MRAN Algorithm

I.J. Rohit*, I. Jacob Raglend and M. Dev Anand*****

ABSTRACT

One of the significant problem in robot kinematics is to optimising the solution of inverse kinematics which deals with obtaining the joint variables in terms of the end-effector position and orientation and is difficult than the forward kinematics problem. As the degree of freedom of a robot increases the inverse kinematics calculation become more difficult and expensive. This paper proposes neural network architecture to optimise the inverse kinematics solution.The neural networks ideasconsidered here are Time Delay and Distributed Time Delay Neural Network Algorithms. This technique causes a decrease in the difficulty and calculations faced when using the traditional methods in robotics. Thus the optimised output is evaluated to ensure the efficiency of this approach. In this paper the Neural Network ideaMinimum Resource Allocation Network algorithm is proposed to solve the inverse kinematics problem of a five degree of freedom robot end effector. This technique causes a decrease in the difficulty and calculations faced when using the traditional methods in robotics. Thus the optimised output is evaluated to ensure the efficiency of this approach.

Keywords: Degrees of Freedom, Inverse Kinematics, MRAN Algorithm

1. INTRODUCTION

Nowadays robots are considered as an indispensable part of modern manufacturing field with their inherent capability of executing complex and risky jobs more efficiently and reliably. A robotic manipulator is composed of several links connected together through joints. Kinematics deals with the geometric motion of a robotic manipulator and the inverse kinematics is considered as the most popular and efficient method of controlling robot arm. Figure 1 gives a view of the general structure of a series manipulator with revolute joints (5 DOF).

The Figure 2 shows a five Degree of Freedom joints related to waist, shoulder, elbow, pitch and roll.

The proposed approach is a strategy that can be implemented to solve the inverse kinematics problems faced in robotics with highest DOF more efficiently.

Researchers proposed a Neuro-Genetic Approach to determine the inverse kinematics solution of robotic manipulators.The proposed solution method is based on using Neural Networks (NN) and Genetic Algorithms (GA) in a hybrid system.Here an Elman NN as well as GA ideas are implemented.The error introduced by the NN can be minimised by the application of GA. The main problems are the test errors and learning time is larger and it requires large number of hidden neurons [1].

Adrian-VasileDuka proposed a Novel Approach on NN Based inverse kinematics solution for trajectory tracking of a robotic arm (2011).It employsthe conventional feed forward NN.By using this idea the desired

* PG Student, Department of Aeronautical Engineering, Noorul Islam Centre for Higher Education, Kumaracoil-629 180, Thuckalay, Kanyakumari District, Tamilnadu State, India, *Email: ijrohit1992@gmail.com*

** Professor and Deputy Director Research, Department of Electrical and Electronics Engineering, Noorul Islam Centre for Higher Education, Kumaracoil-629 180, Thuckalay, Kanyakumari District, Tamilnadu State, India, *Email: jacobraglend@rediffmail.com*

*** Professor and Deputy Director Academic Affairs, Department of Mechanical Engineering, Noorul Islam Centre for Higher Education, Kumaracoil-629 180, Thuckalay, Kanyakumari District, Tamilnadu State, India, *Email: anandpmt@hotmail.com*

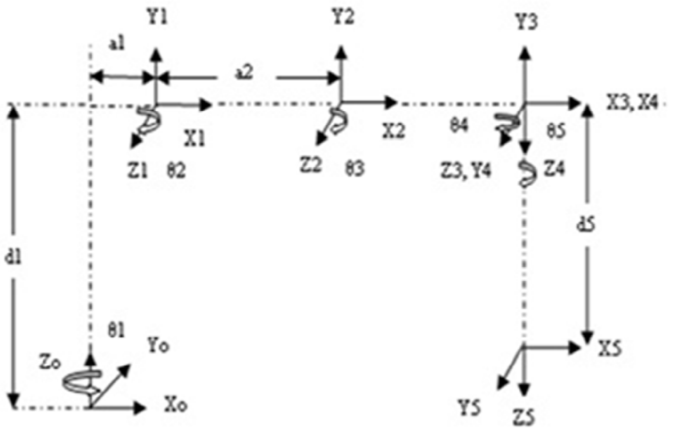


Figure1: Five Joint Manipulator Arm Links and Joints

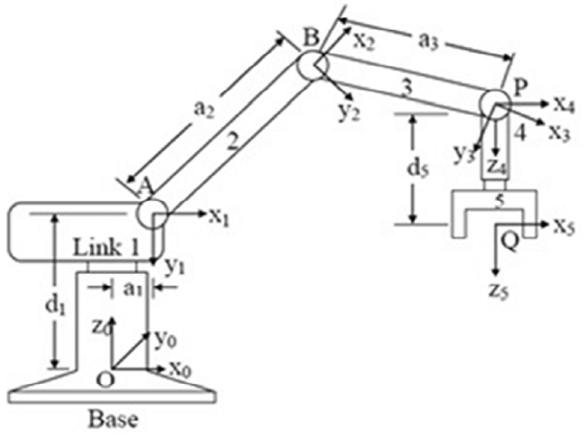


Figure 2: A Five Joint Robot Arm

trajectories can begenerated by solving inverse kinematics problem. Since it employs the conventional method it performs simple operation [2].

Daniela Tarnitaa, Marghitu presented an analysis of a hand arm system (2013.The algorithm is based on the calculation of largest Lyapunovfunction.By calculating theLLE thestability of a dynamical system can be determined. But the calculation of Lyapunov exponent is very complex[3]. Researchers proposed an approach usingNNarchitecture for inverse kinematics problem in robotics.The NN utilized aMulti-Layered Perceptron (MLP) training algorithm using back-propagation. Complexity of the algorithm will be reduced. The applied traditional algorithm is not much effective for complex structures [4].

Harmony Search Algorithm (HSA) was proposed in [5] for robot localization throughscan matching.The HSA has been applied for mobile robot localization and it outperformed the HSA-based approach. HSA has slow convergence speed.

ANNs' based inverse kinematics solution for serial robot manipulators passing through singularities was given in [6].This paper proposes the back propagation algorithm with sigmoid function as an activation function.Since it is a simpler algorithm, it is very easy in calculation and implementation.But it does not have the ability to learn huge number of patterns thus it is limited to small number of data patterns and the error percentage is higher.

J. Ramirez A., and A. Rubiano F proposed an optimization of inverse kinematics of a 3R robotic manipulator using GAs (2011).Under certain conditions, GA is not appropriate to solve inverse kinematics problems in fast and accurate way since it has a higher response time [7].

Researchers introduced a developmental approach to robotic pointing via human–robot interaction (2014).MRAN technique is employed which has an incremental feature that fits the developmental robot very well.It works by first applying a reinforcement learning algorithm to guidethe robot to create attempt movements towards a salient object that is out of the robot's initial reachable space.But it has only less accuracy [8], [12-13].

To optimize the joint angles of robotics manipulator using GA was given in [9]. The system would adopt the advantage of GA to optimize its performance in terms of path control and accuracy. Once a path is being generated and given as input to the robot, the manipulator's end tip moves along that specified path.Thus parameters can be found with high accuracy even with low resolution encoders.But it is a time consuming procedure which is a disadvantage. Similarly the Identification of time-varying nonlinear systems using minimal Radial Basis Function NNs(RBFNN) was given in [10]. An identification algorithm for time varying nonlinear systems using a sequential learning scheme with a minimal RBFNN is presented. The learning algorithm combines the growth criterion of the resource allocating network with a pruning strategy. Researchers proposed antrajectory planning forthe planar redundant manipulator. The goal is to minimize the sum of the end-effector position error at each intermediate point along the trajectory which makes the end-effector to track the prescribed trajectory accurately for a 3DOF planar manipulator with different end-effector trajectories have been carried out [11].

This paper proposes the neural network idea Minimum Resource Allocation Network algorithm, and here the algorithms undergo the growing as well as the pruning method to train the network.

2. FORWARD KINEMATIC MODEL OF SCORBOT-ER VU PLUS INDUSTRIAL ROBOT

The answer for the forward kinematics issue comprises of discovering the estimation of the extreme location of TCP. This result is a capacity of 5 joint qualities, and D-H parameters. There are a few techniques to intensify this issue. This research is carried out utilizing the homogeneous conversion matrices technique, and the D-H's deliberate representation of the reference frameworks. Despite the fact that the last position might be discovered geometrically, the technique proposed in this work offers a reaction which could compare the location of the extreme of each connection in the kinematics network, contrasted with the past or the worldwide standard framework, so as to characterize the position of every explanation in the robot.

2.1. Frame Assignment and Structure

The joints of the mechanical arm of the Scorbot-ER Vu Plus Robot are identified. The D-H parameters as indicated by this model are shown in Table 1. The kinematics model is demonstrated in, with the edge assignments as per the D-H documentations.

2.2. Denavit-Hartenberg Representation

The D-H matrix is a special form of a homogeneous transformation matrix, a 4×4 matrix, having the property of transforming a vector from one coordinate frame to another, by means of a translation or rotation. For a kinematic chain with n-joints and $n - 1$-links, every joint is assigned a frame of reference. Thus, each joint can be represented by a homogeneous transformation matrix, describing the particular rotation or translation needed to align the i^{n-1th}joint with the i^{th}joint. The product of these matrices gives the final position of the n^{th} joint. Additionally, it proposed a methodical documentation for allotting the right united ortho-normal correlate skeleton, every one connection in a chain of open kinematic connections. When these connections appended direction frames are doled out, the conversions between neighbouring coordinate frames could be spoken to by a solitary standard 4x4 homogeneous coordinate transformation matrix.

The coordinates are allotted to the connections utilizing the accompanying methodology.

- Joints number from 1 to n, beginning with the base and finishing with the device yaw, pitch, and roll, in a specific order.
- Allot coordinate frame L_0 of right-handed ortho-normal to the robot, verifying that $z°$ adjusts to the axis of joint 1. Set $k = 1$.

Table 1
D-H Parameters for the Scorbot-ER Vu Plus Robot

Joint i	α_i	a_i	d_i	θ_i	*Operating Range*
1	$-\pi/2$	$a_1 = 10$	$d_1 = 5$	$\theta_1 = 30°$	310°
2	0	$a_2 = 15$	$d_2 = 0$	$\theta_2 = 45°$	+130°/–35°
3	0	$a_3 = 20$	$d_3 = 0$	$\theta_3 = 60°$	±130°
4	$-\pi/2$	$a_4 = 0$	$d_4 = 0$	$\theta_4 = 50°$	±130°
5	0	$a_5 = 0$	$d_5 = 5$	$\theta_5 = 70°$	±570°

- Adjust the axis of joint k + 1 with z^k.
- Locate the source of L_k at the crossing point of the z^k and z^{k-1} axis. On the off chance that they don't converge, use convergence of z^k with a typical ordinary in the middle of z^k and z^{k-1}.
- Choose x^k to be orthogonal to z^k and z^{k-1} both. In the event that z^k and z^{k-1} are parallel, point x^k far from z^{k-1}.
- Choose y^k to structure an ortho-normal coordinate outline L_k.
- Make $k = k + 1$. In the event that $k < n$, go to step 2; else, proceed.
- Make the starting point of L_n at the device tip. Adjust z^n to the methodology vector, y^n the sliding vector, and x^n with the typical vector of the tool. Set $k = 1$.
- Allocate point b^k at the convergence of the x^k and Z^{k-1} pivot. In the event that they don't cross, utilize the crossing point of x^k with a typical ordinary in the middle of x^k and z^{k-1}.
- Find θ^k as the angle of turn from x^{k-1} to x^k measured about z^{k-1}.
- Find d_k as the distance from the beginning of frame L_{k-1} to point b^k, measured along z^{k-1}.
- Find a_k as the distance from point b^k to the beginning of frame L_k, measured along x^k.
- Find α_k as the angle of turn from z^{k-1} to z^k measured about x^k.
- Set $k = k + 1$. In the event that $k \Leftarrow n$, go to step 8; else, stop.

2.3. Transformation Matrix

In the wake of creating the D-H coordinate framework for every connection, a similar matrix of transformation can without much of a stretch be produced, bearing in mind body{i-1} and body{i} change comprising of four essential conversions. The general complex homogeneous matrix of transformation can be shaped by sequential applications of basic changes. This transformation comprises of four essential conversions.

T$_1$: Angle θ_i for z_{i-1} axis of rotation

T$_2$: Distance d_i for z_{i-1} axis of translation

T$_3$: Distance a_i along x_i axis of translation and

T$_4$: Angle α_i about x_i axis of rotation

Taking into account the D-H gathering, the transformation matrix from joint i to joint $i+1$ is prearranged by:

$$^{i-1}T_i = \begin{bmatrix} C\theta i & -S\theta iC\alpha i & S\theta iS\alpha i & aiC\theta i \\ S\theta i & C\theta iC\alpha i & -C\theta iS\alpha i & aiS\theta i \\ 0 & S\alpha i & C\alpha i & di \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Where $S\theta_i$ = Sin θ_i, $C\theta_i$ = Cos θ_i, $S\alpha_i$ = Sin α_i, $C\alpha_i$ = Cos α_i. The Overall Transformation Matrix,

$$^0T_5 = {}^0T_1 * {}^1T_2 * {}^2T_3 * {}^3T_4 * {}^4T_5 \tag{2}$$

$$^0T_1 = \begin{bmatrix} C_1 & 0 & -S_1 & a_1C_1 \\ S_1 & 0 & C_1 & a_1S_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & \frac{1}{\pi} \end{bmatrix}$$

Substitute $\theta_1 = 30°$, $\alpha_1 = -\overset{\underline{1}}{\frac{\pi}{2}}$, $a_1 = 10$, $d_1 = 5$ (3)

$$^{0}T_1 = \begin{bmatrix} 0.866 & 0 & -0.5 & 8.660 \\ 0.5 & 0 & 0.866 & 5 \\ 0 & -1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$^{1}T_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Substitute $\theta_2 = 45°$, $\alpha_2 = 0.$ $a_2 = 15$, $d_2 = 0$ (5)

$$^{1}T_2 = \begin{bmatrix} 0.7071 & -0.7071 & 0 & 10.6066 \\ 0.7071 & 0.7071 & 0 & 10.6066 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

$$^{2}T_3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Substitute $\theta_3 = 60°$, $\alpha_3 = a_3 = 20$, $d_3 = 0$ (7)

$$^{2}T_3 = \begin{bmatrix} 0.5 & -0.8660 & 0 & 10 \\ 0.8660 & 0.5 & 0 & 17.3205 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

$$^{3}T_4 = \begin{bmatrix} C_4 & 0 & S_4 & 0 \\ S_4 & 0 & -C_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Substitute $\theta_4 = 50°$, $\alpha_4 = -\frac{\pi}{2}$, $a_4 = 0$, $d_4 = 0$ (9)

$$^{3}T_{4} = \begin{bmatrix} 0.6428 & 0 & -0.7660 & 0 \\ 0.7660 & 0 & 0.6428 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

$$^{4}T_{5} = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Substitute $\theta_5 = 70°$, $\alpha_5 = a_5 = 0$, $d_5 = 5$ (11)

$$^{4}T_{5} = \begin{bmatrix} 0.3420 & -0.9397 & 0 & 0 \\ 0.9397 & 0.3420 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

We know that,

$$^{0}T_{2} = {}^{0}T_{1}{}^{*1}T_{2} = \begin{bmatrix} 0.6123 & -0.6123 & -0.5 & 17.8453 \\ 0.3536 & -0.3536 & 0.866 & 10.3033 \\ -0.7071 & -0.7071 & 0 & -5.6066 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

$$^{0}T_{3} = {}^{0}T_{2}{}^{*2}T_{3} = \begin{bmatrix} -0.2241 & -0.8364 & -0.5 & 13.3630 \\ -0.1294 & -0.4830 & 0.866 & 7.7148 \\ -0.9659 & 0.2587 & 0 & -24.9249 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}$$

$$^{0}T_{4} = {}^{0}T_{3}{}^{*3}T_{4} = \begin{bmatrix} -0.7847 & -0.5 & -0.3660 & 13.3630 \\ -0.4532 & -0.8660 & 0.2114 & 7.7148 \\ -0.4227 & 0 & 0.9062 & -24.9249 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

$$^{0}T_{5} = T_{e} = {}^{*4}T_{5} = \begin{bmatrix} 0.2015 & 0.9084 & -0.3660 & 11.533 \\ -0.9688 & 0.1297 & -0.2114 & 6.6578 \\ -0.1446 & 0.3972 & 0.9062 & -20.3939 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{16}$$

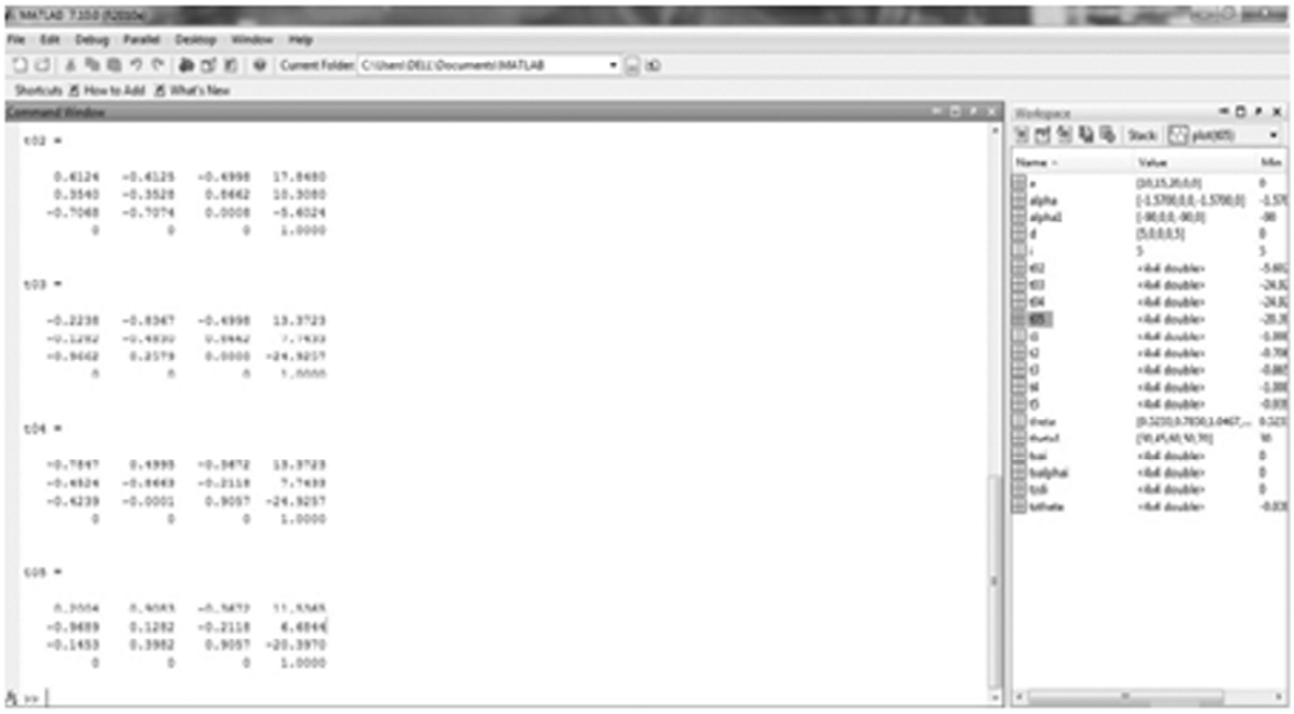
2.4. VERIFICATION OF THE MODEL BY MATLAB

MATLAB is a capable environment for direct arithmetical and graphical depiction that is accessible on an extensive variety of machine stages. The center usefulness could be stretched out by application particular toolboxes. The Simulink tool kit gives numerous capacities that are needed in robotics, and locations ranges, for example, kinematics, dynamics, and path creation. This Tool is valuable for dissection and also investigating results from tries different things with true robots, and could be an influential device for instruction. The Toolbox is focused around an extremely common system for speaking to the kinematics and dynamics of serial-connection controllers, by depiction matrices. These involve, in the least difficult case, the D-H parameters of the robot, and could be made by the client for any serial-link controller. The controller portrayal could be explained, by increasing the matrix, to incorporate connection inertial, and motor inertial, and frictional parameters. Such frameworks give a succinct method for depicting a robot model, and may encourage the imparting of robot models over the exploration group. This would permit the simulation results to be looked at in a considerably more compelling path than is right now done in the writing. The Toolbox additionally gives capacities to controlling information sorts, for example, vectors, homogeneous changes and unit-quaternion, which are important to speak to a 3D position and introduction. The schedules are by and large composed in a direct, or course book, way for pedagogical reasons, instead of for most extreme computational proficiency. With the forward and reverse kinematics for a controller tackled, these equations could be utilized to discover the inputs (joint angles) important to plot the robot. A trajectory could be characterized regarding a tool space move, say, a consecutive-line move of the end-effector, and afterward invigorate the robot over this move. A case is indicated underneath, with a simulated screen of the MATLAB system yield. Figure 3 speaks to the MATLAB result yield document for the forward kinematic.

Knowing the following values:

$a_1 = 10$ mm; $a_2 = 15$ mm; $a_3 = 20$ mm; $d_1 = 5$ mm; $d_5 = 5$ mm; $\theta_1 = 30°$; $\theta_2 = 45°$; $\theta_3 = 60°$; $\theta_4 = 50°$; $\theta_5 = 70°$;

Case Study: (Mathematical Solution)



```
t02 =

    0.4124   -0.4125   -0.4998   17.8480
    0.3540   -0.3528    0.8442   10.3080
   -0.7068   -0.7074    0.0008   -5.4024
         0         0         0    1.0000


t03 =

   -0.2238   -0.8367   -0.4998   13.3723
   -0.1282   -0.4830    0.8442    7.7433
   -0.9662    0.2579    0.0000  -24.9257
         0         0         0    1.0000


t04 =

   -0.7847    0.4995   -0.3672   13.3723
   -0.4524   -0.8443   -0.2118    7.7433
   -0.4239   -0.0001    0.9057  -24.9257
         0         0         0    1.0000


t05 =

    0.2504    0.9083   -0.3472   11.5565
   -0.9489    0.1282   -0.2118    6.6844
   -0.1453    0.3982    0.9057  -20.3970
         0         0         0    1.0000

>>
```

Figure 3: MATLAB Result for the Forward Kinematic

$$^{0}T_{5} = \begin{bmatrix} 0.2015 & 0.9084 & -0.3660 & 11.5330 \\ -0.9688 & 0.1297 & -0.2114 & 6.6578 \\ -0.1446 & 0.3972 & 0.9062 & -20.3939 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

Case-Study: (MATLAB Program Output)

From Table 3.1 Joint Parameter details,

The final matrix ${}_{0}T^{5}$ result:

$$^{0}T_{5} = \begin{bmatrix} 0.2004 & 0.9083 & -0.3672 & 11.5365 \\ -0.9689 & 0.1282 & -0.2118 & 6.6844 \\ -0.1453 & 0.3982 & 0.9057 & -20.3970 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$$

The Final values of matrix (${}_{0}T^{5}$) are compared with the physical locations of the robot arm in Table 2.

2.5. Inverse Kinematic Model

The inverse kinematics is also done on the robot. In this stage, the inverse kinematics formulae are generated using the inverse matrices, and a model is validated and verified, using ROBOCELL and MATLAB. This chapter explains how the inverse kinematics Scorbot-ER Vu Plus Robot mathematical model is created effectively.

2.6. Inverse Kinematics of Scorbot-ER Vu Plus Industrial Robot Manipulator

By placing the $\frac{1}{4}T$ to the other side of the equation, we can separate θ_1 to make easier for its computation as follows:

$$\left[{}^{0}_{1}T\right]^{-1}\left[P\right] = \left[{}^{1}_{4}T\right] \tag{17}$$

$$P = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

The inverse transformation matrix can be computed by using the upcoming matrix.

Table 2
Differences between the Analytical and Physical Values of the Robot (Forward Kinematic)

Position Values	*Analytical ${}_{0}T^{5}$ Values (mm)*	*Matlab ${}_{0}T^{5}$Values (mm)*	*Difference (mm)*	*Analytical ${}_{0}T^{5}$ Values (mm)*	*Robo cell ${}_{0}T^{5}$ Values (mm)*	*Difference (mm)*
P_x	11.5330	11.5365	0.0035	11.5330	11.4665	0.0665
P_y	6.6578	6.6844	0.0266	6.6578	6.6711	0.0133
P_z	–20.3939	–20.3970	0.0031	–20.3939	–20.395	0.0015

$$[T]^{-1} = \left(\begin{array}{c|c} R^T & -R^T\ P \\ \hline 0 & 1 \end{array} \right) \tag{19}$$

Applying this to yields the following result,

$$\left[{}^0_1T\right]^{-1} = \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

Therefore,

$$\begin{bmatrix} C_1 & S_1 & 0 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_{23} & -S_{23} & 0 & a_1 + a_2C_2 + a_3C_{23} \\ -S_1 & C_1 & 0 & d_2 \\ 0 & 0 & 1 & -a_2S_2 + a_3S_{23} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

For compute θ_1 the element (2, 4) in the matrix can be utilized.

$$d_2 = -\sin(\theta_1)\, p_x + \cos(\theta_1)\, p_y \tag{22}$$

Letting,

$$p_x = \rho \cos \tag{23}$$

$$p_y = \rho \sin \tag{24}$$

Where,

$$\rho = \sqrt{p_x^2 + p_y^2} \tag{25}$$

ϕ = Arctan 2 ($p_{y,}\ p_x$) (26)

We know that,

$$\frac{d_2}{\rho} = -\sin(\theta_1)\cos(\phi) + \cos(\phi_1)\sin\phi \tag{27}$$

Therefore,

$$\sin(\phi - \theta_1) = \frac{d_2}{\rho} \tag{28}$$

$$\cos(\phi - \theta_1) = \pm\sqrt{1 - \frac{d_2^2}{\rho^2}} \tag{29}$$

$$\phi - \theta_1 = \arctan 2\left(\frac{d_2}{\rho}, \pm\sqrt{1 - \frac{d_2^2}{\rho^2}} \right) \tag{30}$$

This leaves the solution for θ_1 as shown in,

$$\theta_1 = \arctan 2\,(p_{y,}\, p_x) \text{ -arctan } 2\left(d_2,\ \pm\ \sqrt{p_x^2 + p_y^2 - d_2^2}\right) \tag{31}$$

For compute the joint angle three, we must verify the elements from (1, 4) and (3, 4). First,

$$C_1\, p_x + S_1\, p_y = a_3\, C_{23} + a_2\, C_2 + a_1 \tag{32}$$

$$p_z - d_1 = -a_3\, S_{23} - a_2\, S_2 \tag{33}$$

Next by squaring Equation (32) and Equation (33) followed by addition, θ_3 can be determined.

$$(C_1\, p_x + S_1\, p_y - a_1)^2 + (d_1 - p_z)^2 = (a_3\, C_{23} + a_2\, C_2)^2 + (a_3\, S_{23} + a_2\, S_2)^2 \tag{34}$$

$$(C_1\, p_x + S_1\, p_y - a_1)^2 + (d_1 - p_z)^2 = a_2^2 + a_3^2 + 2a_2\ a_3(C_2\ C_{23} + S_2\ S_{23}) \tag{35}$$

$$(C_1\, p_x + S_1\, p_y - a_1)^2 + (d_1 - p_z)^2 = a_2^2 + a_3^2 + 2a_2\ a_3\ C_3 \tag{36}$$

This leaves Equation (3.37) for $\theta_{3,}$

$$\theta_3 = \pm\arccos\left(\frac{\left(C_1 p_x + S_1 p_y - a_1\right)^2 + \left(d_1 - p_z\right)^2 - a_2^2 - a_3^2}{2a_2 a_3}\right) \tag{37}$$

Next by transferring the matrix ${}^1_2T$ onto the other side, next equation can be found that will permit us to find θ_2.

$$\left[{}^1_2T\right]^{-1}\left[{}^0_1T\right]^{-1}[P] = \left[{}^2_4T\right] \tag{38}$$

$$\left[{}^1_2T\right]^{-1} = \begin{bmatrix} C_2 & 0 & -S_2 & -a_1C_2 \\ -S_2 & 0 & -C_2 & a_2S_2 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{39}$$

$$\left[{}^1_2T\right]^{-1}\left[{}^0_1T\right]^{-1} = \begin{bmatrix} C_2 & 0 & -S_2 & -a_1C_2 \\ -S_2 & 0 & -C_2 & a_1S_2 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} C_1 & S_1 & 0 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

$$\left[{}^1_2T\right]^{-1}\left[{}^0_1T\right]^{-1} = \begin{bmatrix} C_1C_2 & S_1C_2 & -S_2 & S_2d_1 - a_1C_2 \\ -C_1S_2 & -S_1S_2 & -C_2 & C_2d_1 + a_1S_2 \\ -S_1 & C_1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{41}$$

$$\begin{bmatrix} C_1C_2 & S_1C_2 & -S_2 & S_2d_1 - a_1C_2 \\ -C_1S_2 & -S_1S_2 & -C_2 & C_2d_1 + a_1S_2 \\ -S_1 & C_1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_3 & -S_3 & 0 & a_3C_3 + a_2 \\ S_3 & C_3 & 0 & a_3S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{42}$$

By finding elements (1, 4) of the above Equation (42) we stick with the following formula.

$$C_1 C_2 p_x + S_1 C_2 \; p_y - S_2 p_z + d_1 S_2 - C_2 a_1 = a_3 C_3 + a_2 \tag{43}$$

$$C_2 (C_1 p_x + S_1 p_y - a_1) + S_2 (d_1 - p_z) = a_3 C_3 + a_2 \tag{44}$$

Substituting $A = (C_1 p_x + S_1 p_y - a_1)$ $B = (d_1 - p_z)$ and $C = a_3 C_3 + a_2$ it is easy to solve θ_2 via reduction to a polynomial.

$$C_2 A + S_2 B = C \tag{45}$$

Where,

$$C_2 = \frac{1-U^2}{1+U^2} \tag{46}$$

$$S_2 = \frac{2U}{1+U^2} \tag{47}$$

Substituting the above two Equation (3.46) and Equation (3.47) into Equation (3.45) and rearranging, Equation (3.48) is obtained.

$$(C + A)U^2 - 2UB + (C - A) \tag{48}$$

The quadratic formula rendering can be used to solve this:

$$U = \frac{B \pm \sqrt{B^2 + A^2 - C^2}}{A + C} \tag{49}$$

Where $\rightarrow$

$$\theta_2 = 2 \arctan (U) \tag{50}$$

The angle of the 4th joint, θ_4, can also be simply determined, based on θ_2 and θ_3.

$$p_z = p_z' - a_4 \tag{51}$$

$$\theta_4 = 90 - \theta_2 - \theta_3 \tag{52}$$

Replace the *z* offset in Equation (3.49), we find the upcoming set of formulae for the inverse kinematics of the *Scorbot-ER Vu Plus Robot.*

$$\theta_1 = arctan\ 2\ (p_{y,}\ p_x)\ arctan\ 2 \left(d_2,\ \pm \sqrt{p_x^2 + p_y^2 - d_2^2} \right) \tag{53}$$

$$\theta_3 = \pm arccos \left(\frac{\left(C_1 p_x + S_1 p_y - a_1\right)^2 + \left(d_1 - p_z' + a_4\right)^2 - a_2^2 - a_3^2}{2 a_2 a_3} \right) \tag{54}$$

$$\theta_2 = 2arcta \left(\frac{\left(d_1 - p_z' + a_4\right) \pm \sqrt{\left(d_1 - p_z' + a_4\right)^2 \left(c_1 p_x + S_1 p_y - a_1\right)^2} - \left(a_3 C_3 + a_2\right)^2}{\left(c_1 p_x + S_1 p_y - a_1\right) + \left(a_3 c_3 + a_2\right)} \right) \tag{55}$$

$$\theta_4 = 90 - \theta_2 - \theta_3 \tag{56}$$

Although in this case θ_5 was never changed, by substituting all the values of Table 3.1, in the above formulae, the values can be found.

2.7. Verification of the Model by MATLAB and ROBOCELL

ROBOCELL Software is used to validate the Inverse Kinematic Model of the Scorbot-ER Vu Plus Industrial Robot.

2.7.1. Components of ROBOCELL

ROBOCELL software coordinates the four segments:

i. SCORBASE, full-emphasized robotics control programming software, which gives an easy to use device for robot programming and process.

ii. A module with graphic exhibit that provides robot's 3D dissection of the robot and different gadgets in a fundamental robotic environment, where one can characterize (instruct) the robot movements and accomplish robot programs.

iii. Cell setup, which permits a user to make another virtual automated workcell, or change a current workcell.

iv. 3D simulation software display to exhibit ROBOCELL's capacities.

v. ROBOCELL's illustration of a robot and gadgets is focused around the genuine measurements and capacities of the Scorbot-ER Vu Plus Robot equipment. In this way, working and programming the robot in ROBOCELL could be utilized with a genuine robotic establishment. Realistic presentation peculiarities and programmed operations, for example, cell reset and send robot commands, empower fast and exact programming. SCORBASE's imitate the client interface and menus of ROBOCELL. SCORBASE operations, menus and commands are depicted in the SCORBASE hand book.

vi. ROBOCELL gives the strategies depicted beneath, for characterizing the positions of the robot. Allotted number indicates its position.

2.7.2. Recording Position (First Method)

i. The virtual robot is controlled by the SCORBASE physical dialog box like a real robot.

ii. In the teach position (Simple) dialog box a number is written in the location number field when the position is reached.

iii. Click the record button.

iv. The new position will replace the existence position data in the event that the position number has been utilized long ago.

2.7.3. Recording Position (Second Method)

i. Position and above position tools are used for specific location of robot, while robot moves to object.

ii. Calibration is done by utilizing the manual movement dialog box.

iii. Until the position is reached, location number field's in the teach position (Simple) dialog box is written.

iv. Click the record button.

v. In the event that the position number has been utilized at one time, the new position will overwrite the past position information.

2.7.4. *Teaching Position*

i. Object X and Y coordinates notice in the Graphic Display window, the selection has been made on “View/Show Positions”.

ii. To record the coordinates of the object or point, it should be zoomed in.

iii. To open the Teach Position dialog box (Figure 4) the “Teach Position (Simple) Expanded button” is clicked.

iv. Position coordinates X, Y, Z, P, and R are entered in their respective fields.

v. The position number field is entered with a number.

vi. The teach button is clicked.

In the event that the “Record Position Catch” is clicked, the location of previous robot is taken (and not the locations characterized by the directions imported in the X, Y, Z, P, and R fields).

The new position will replace the previous position datas when the position number has been used formerly.

2.7.5. Fine-Tuning a Position

To modify the existing positions:

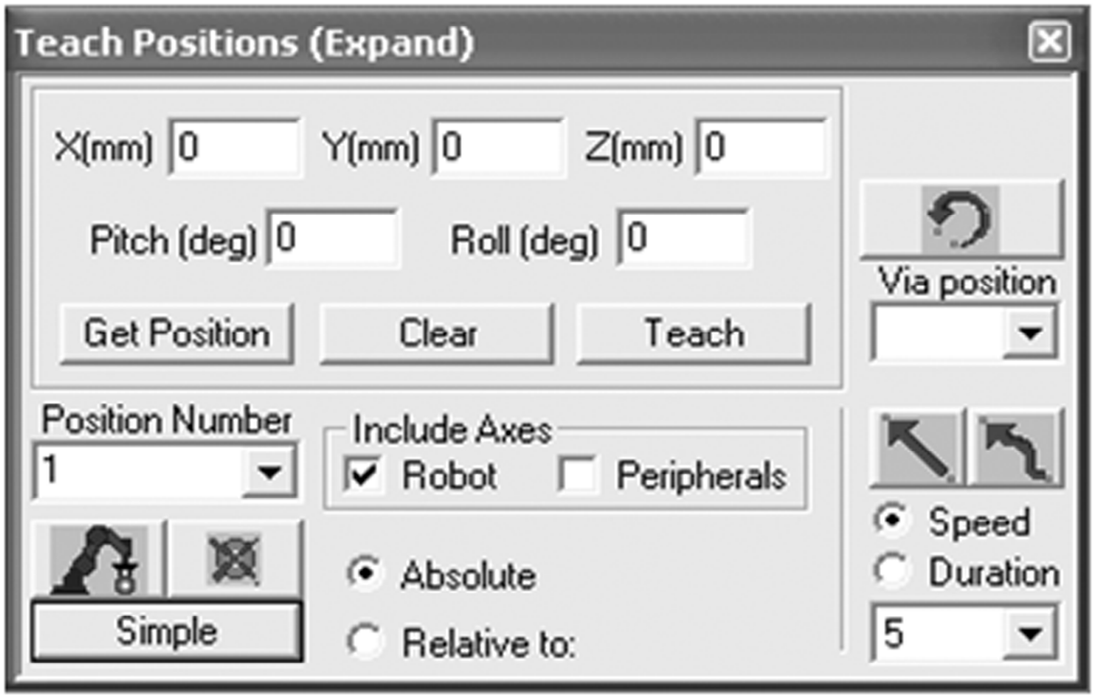
i. The “Teach Position (Simple) Expanded button” is selected to unwrap the Teach Position dialog box.

ii. To modify a position, it is selected in the “Position Number Field”.

iii. The “Get Position” button is clicked. The position data presents in the fields of X, Y, Z, P, and R.

iv. The required coordinate is modified.

v. To overwrite the previous position, the “Teach button” is clicked.


Teach Positions (Expand)

X(mm) 0 Y(mm) 0 Z(mm) 0

Pitch (deg) 0 Roll (deg) 0

Position Number 1

Include Axes: [✓ Robot] Peripherals

[● Absolute] Relative to:

[● Speed] Duration

5

Figure 4: Teach Position (Expand)

2.7.6. Program Execution

ROBOCELL project execution is the similar as performing project while utilizing a genuine robot framework. Diverse cell arrangements might be stacked and changed in ROBOCELL. Be that as it may the positions and projects are not stacked together with their work cell. For another task, to consider the work cell and its positions, the Save as choice in the File menu might be utilized. The undertaking with work cell and positions is spared under an alternate name. At that point the project is erased and another one is composed. (The positions and the cell stay unaltered).

For a given set of parameters (Table 1), a program in MATLAB 7.10 and ROBOCELL is created, and the comparison between its output with the experimental output as mentioned below. See Figure5 and 6. Table 3 shows the differences between the Analytical and physical values of the robot.

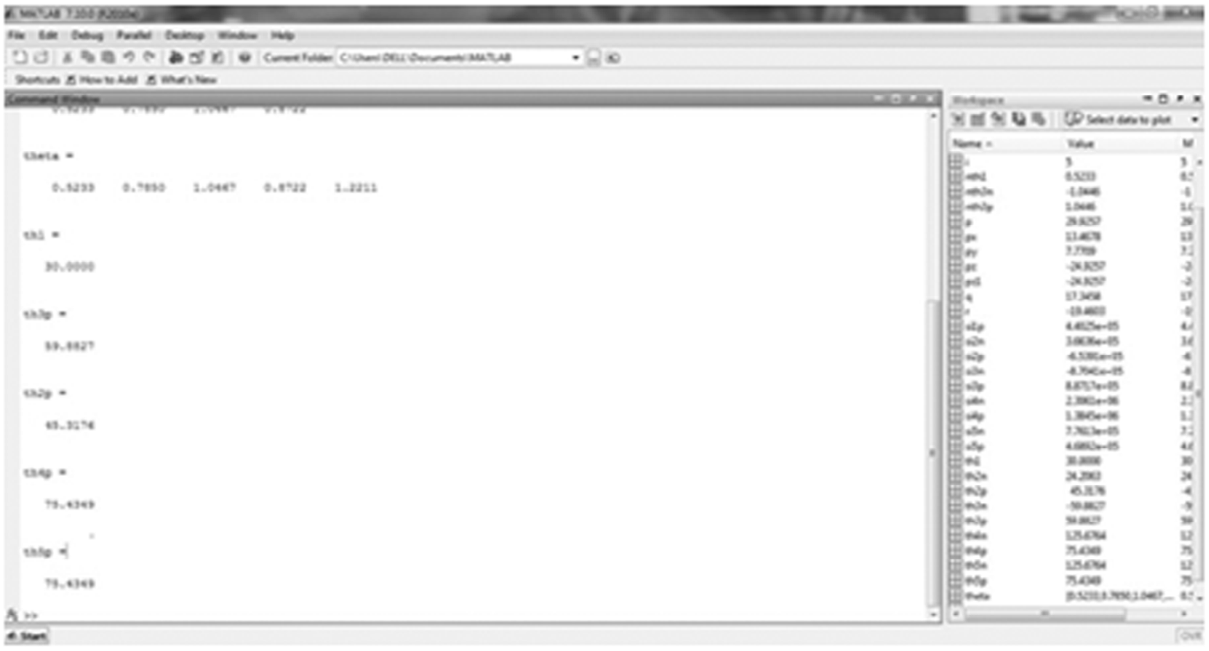
3. METHODOLOGY

3.1. Artificial Neural Network

It is a computational system influenced by the structure, processing method and learning ability of a biological brain. The characteristics of an ANN include a large number of very simple processing neuron like processing elements, a large number of weighted connections between the elements and the distributed representation of knowledge over the connections. This knowledge is acquired by network through a learning process.

Table 3
Differences between the Analytical and Physical Values of the Robot (Inverse Kinematic)

Position Values	*Analytical θ Values (°)*	*Matlab θ Values (°)*	*Error*	*Analytical θ Values (°)*	*Robo cell θ Values (°)*	*Error*
θ_1	30°	30°	0°	30°	30.16°	0.16°
θ_2	45°	45.31°	0.31°	45°	45.27°	0.27°
θ_3	60°	59.88°	0.12°	60°	60.25°	0.25°
θ_4	75°	75.43°	0.43°	75°	75.11°	0.11°
θ_5	75°	75.43°	0.43°	75°	75.11°	0.11°



```
theta =

    0.5233    0.7850    1.0467    0.8722    1.2211


th1 =

   30.0000


th3p =

   59.8827


th2p =

   45.3176


th4p =

   75.4349


th5p =

   75.4349
```

Name	Value
i	5
nth1	0.5233
nth2n	-1.0446
nth2p	1.0446
p	29.8257
px	13.4678
py	7.7709
pz	-24.8257
pz1	-24.8257
q	17.3458
r	-19.4603
s1p	4.4525e-05
s2n	3.6636e-05
s2p	-6.5391e-05
s3n	-8.7041e-05
s3p	8.8717e-05
s4n	2.3902e-06
s4p	1.3845e-06
s5n	7.7613e-05
s5p	4.6892e-05
th1	30.0000
th2n	24.2063
th2p	45.3176
th3n	-59.8827
th3p	59.8827
th4n	125.6764
th4p	75.4349
th5n	125.6764
th5p	75.4349
theta	[0.5233,0.7850,1.0467,...

Figure 5: MATLAB Result for the Inverse Kinematic

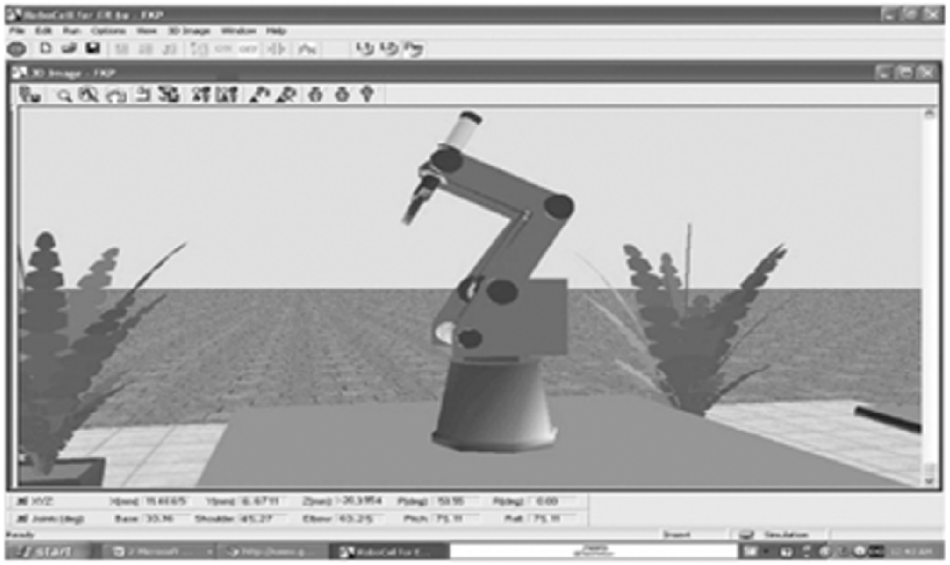


Figure 6: ROBOCELL Result for the Inverse Kinematic

3.2. Minimal Resource Allocation Network (MRAN)

MRAN is minimal radial basis function which is mostly employed in sequential data and it provide this RBF network only by adding and removing the hidden neurons based on the input and can be adopted for online adaptive control of time varying nonlinear system The three main steps of MRAN network are (i). Basic error computing (ii). Criteria for adding hidden units (iii). criteria for adding pruning units. Learning with a fixed size of network will be vague. The networks learning can be increased by increasing or decreasing the network size. The flow chart for the proposed algorithm is given in Figure 7.

3.2.1. Basic Error Computing

For each input $x(i)$, compute:

$$\phi_k(x(i)) = \exp\left(\frac{1}{\sigma_k^2}\|x(i)-\mu_k\|^2\right) \tag{57}$$

$$\hat{y}(i) = f(x(i), i) = \alpha_0 + \textstyle\sum_{k=1}^{N_k} \alpha_K \phi_K(x(i)), \tag{58}$$

$$\in_i = \max\{\varepsilon_{max}\gamma^i, \in_{min}\}, (0 < \gamma < 1) \tag{59}$$

$$e(i) = y(i) - \hat{y}(i), \tag{60}$$

$$e_{rmsi} = \sqrt{\frac{\sum_{j=i-(nw-1)}^{i}[y(j)-\hat{y}(j)]^T[y(j)-\hat{y}(j)]}{nw}} \tag{61}$$

In the MRAN algorithm, when the network begins there will be zero hidden neurons. As certain training steps over, the modifications in the network is built up based on certain growing method.It is achieved only by satisfying certain conditions. The algorithm adds or prunes hidden units as well as adjusting the existing network parameters to maintain the input output behaviour. A brief outline of the steps like adding and pruning in the MRAN is given below.

3.2.2. Steps for Adding Hidden Units

Step 1 Collect the input X, and compute the result

Step 2 Generate the new hidden unit if and only if the successive conditions are satisfied

- The calculated error should be more than the threshold value
- For the past data compute the RMS Value of the error and it should beats the threshold
- The distance between the centres and the neurons should be a significant value.

3.2.3. Criteria for Growing

If $$\left\| e^{i} \right\| > e_{\min} \tag{63}$$

$$\left\| x(i) - \mu_{nr} \right\| > \in_{i} \tag{64}$$

And $$e_{rmsi} > e_{min}, \tag{65}$$

Then add a fresh hidden unit with;

$$\alpha_{N_h} = e(i) \tag{66}$$

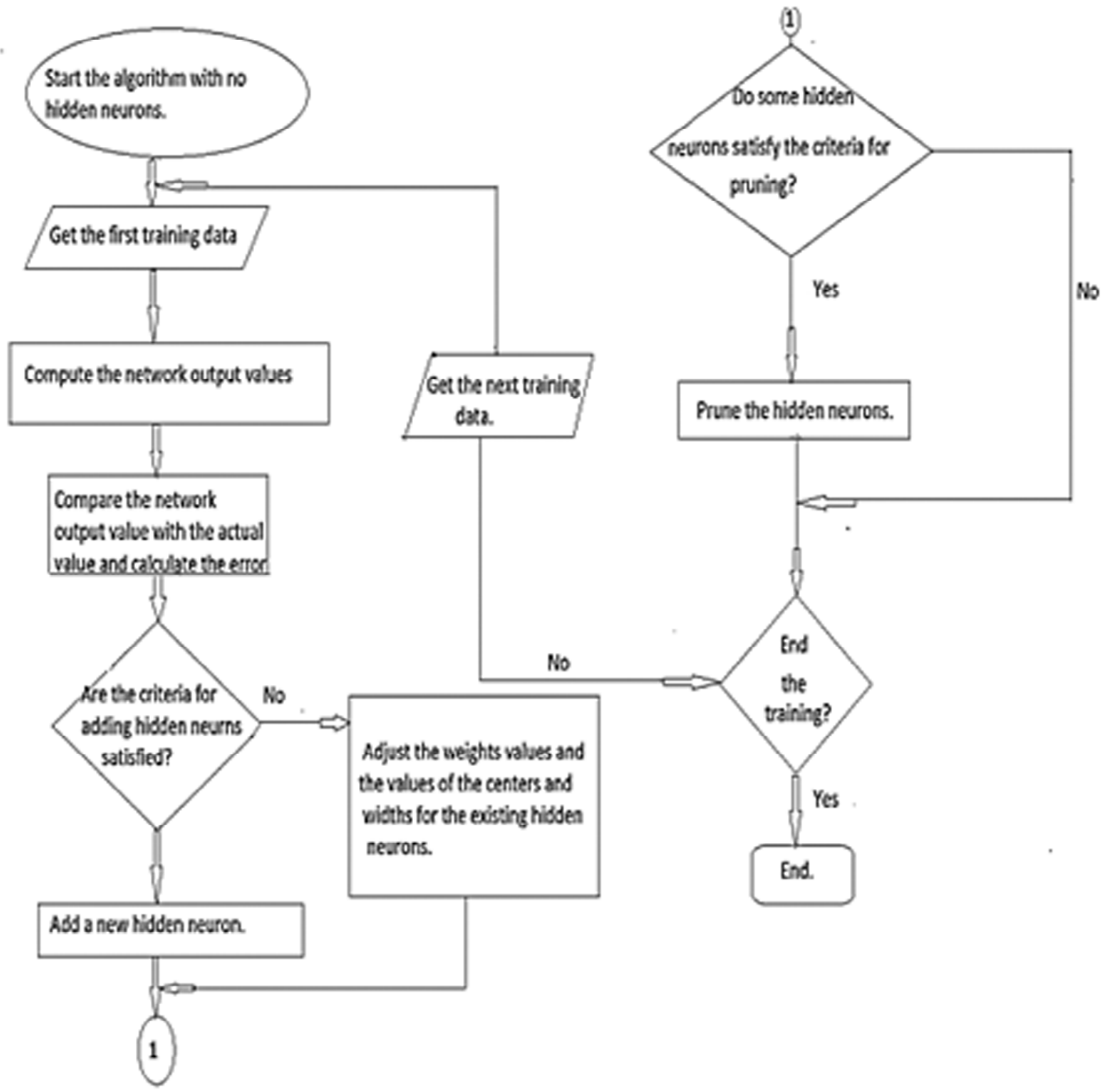

Start the algorithm with no hidden neurons.
Get the first training data
Compute the network output values
Compare the network output value with the actual value and calculate the error
Are the criteria for adding hidden neurns satisfied?
No
Add a new hidden neuron.
Adjust the weights values and the values of the centers and widths for the existing hidden neurons.
1
Get the next training data.
No
1
Do some hidden neurons satisfy the criteria for pruning?
Yes
No
Prune the hidden neurons.
End the training?
Yes
End.

Figure 7: Flow Chart for MRAN Architecture

$$\mu_{N_h} = x(i) \tag{67}$$

$$\sigma_{N_h} = k\left\| x(i) - \mu_{nr} \right\| \tag{68}$$

Else

$$w_i = w_{i-1} + K_i e(i) \tag{69}$$

$$K_i = P_{i-1}A_i + \left[V + A_i^T T P_{i-1} A_i\right]^{-1}, \tag{70}$$

$$P_i = \left[I - K_i A_i^T\right] P_{i-1} + q_0 I \tag{71}$$

Step-3. If the conditions in Step 2 are not met, adjust the weights and widths of the existing RBF network using an extended KalmanFilter (EKF).

3.2.4. Steps for Adding Pruning Units

Pruning means the least useful nodes and weights are iteratively removed and adjusted to maintain the original *i*/*p* *o*/*p* behavior. Some of the steps involved in pruning strategy is given as;

For every observation (X_n, Yn), compute the outputs of hidden unit so ${}^{i}_{k}$, ($k = 1, \ldots\ldots, N_h$)

Find the largest absolute hidden unit output value and Compute the normalized output values for all hidden units

$$\left\| o^i_{I,\max} \right\| = \left[\left\| o^i_{I,\max} \right\|, \ldots, \left\| o^i_{I,\max} \right\|, \ldots, \left\| o^i_{P,\max} \right\| \right]^T \tag{72}$$

Calculate the normalized output vector γ^i_k for each hidden unit in which the k^{th} element is expressed as

$$\gamma^i_k = \left\| \frac{o^i_{IK}}{o^i_{I,\max}} \right\|, \ \left(K = 1, \ \ldots\ldots \ N_h, \ I = 1\ldots p\right) \tag{73}$$

If the values of all elements in γ^i_k are less then δ for the consecutive observations, then prune the k hidden units.

The flow chart makes us understand the MRAN algorithm clearly. The algorithm begins with no hidden neurons. As time elapses the hidden units are added and are based on certain conditions. After starting the network, the training data is obtained and compute the network output values aswell as the error between the actual value and the obtained value.Then we put forward certain criteria and if its satisfies the addition of new neuron takes place. If some neurons satisfies the pruning criteria the removing of hidden units takes place. Thus by adding and pruning the adjustments can be made to maintain the input output behaviour.

4. RESULT AND DISCUSSION

The positions and orientation corresponding to the Cartesian coordinate system has been analyzed by employing the MRAN algorithms by using a set of joint angles as targets. The network got trained using the given input parameter values and can be simulated using MATLAB software. The mean square error estimation by the program also gives the performance and later it can be analyzed. The performance is plotted by the function mean square error across the number of iterations.

Since considering the inverse kinematics, in this work, the Cartesian coordinates X (150,320), Y (0,150) and Z (-24,505) are given as inputs and θ_1(0,45), θ_2(-120,43), θ_3(20,108), θ_4(-3,89), θ_5(0,70) are the angles between the inputs and joints which is called target angles. There are three inputs and five outputs and the goal is to obtain a minimum mean square error which means a performance algorithm. The positions and orientation based on the Cartesian coordinate system has been analysed by employing the artificial neural network method, MRAN by using a set of joint angles as desired output. The network got trained using different parameter values and can be simulated using MATLAB software. The program also gives the mean square error by which the performance can be analysed.

The output of MRAN algorithm is obtained as histogramand is given in the graphical representation of thedistributionof numerical data in Figure 8. In histogram representation the first step is dividing the input data range into certain classes known as bins. Then the number of data points that includes in each and every bin is counted. Thus the height each bar in a histogram is proportional to the data in each class i.e. Count.

In this diagram five colours can be seen and each colour represents the joint angles ie., since a 5 DOF robot is considered there are five joint angles as output responses which is shown in five different colours. The initial step is splitting the data ranges in to bins and each bins may contain one joint angle data or more than one angle data. The maximum height of a bar represents the maximum data values lie in that bin. Here theFrequencyis denoted in the Vertical axis (i.e., counts obtained from each bin) and the Response variable is shown in the Horizontal axis.

The values of certain parameters that considered in the MRAN algorithm is mentioned Table 4. The parameters considered are MSE, Skewness and Kurtosis. Mean Square Error represents the error between the actual and obtained output. Skewness is the measure of asymmetry. For symmetric graph, the skewness

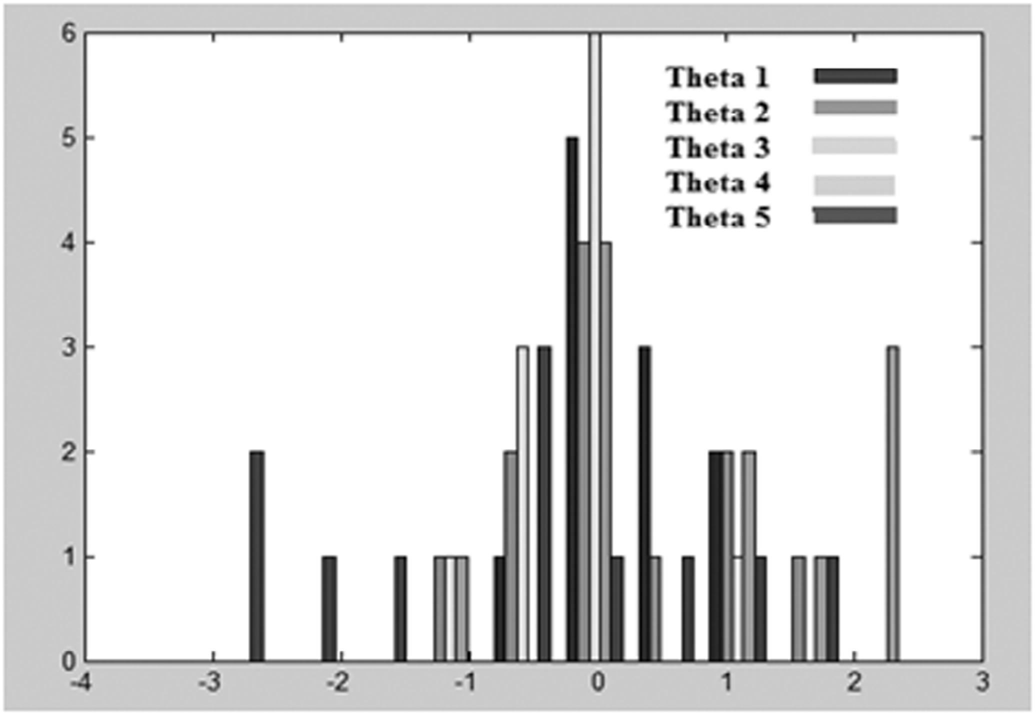


Figure 8: Histogram Representation of MRAN Output

Table 4
Values Obtained for the MRAN Parameters

Parameters	*1*	*2*	*3*	*4*	*5*
MSE	0.2913	0.6961	0.2981	2.1209	2.7887
Skewness	0.2043	0.4418	0.5231	-0.0688	-0.1463
Kurtosis	1.7415	2.6637	3.3338	1.7197	1.7990

value will be zero. If the value will be positive, then it shows the distribution extents more to the right half of the graph and negative skewness shows the distributions more in the left half. Kurtosis is the factor which shows the measure of peaked values.

5. CONCLUSION

This work formulates and simulates the solution for the inverse kinematic problem of a five DOF robot is obtained using the intelligent technique, ANN and can be analyzed by the performances of the adopted MRAN algorithm. For improving the accuracy of the predicted joint and link angles, the number of input patterns of various coordinate values to be trained can also be increased.

REFERENCES

[1] Rasit Koker, “A Neuro-Genetic Approach to the Inverse Kinematics Solution of Robotic Manipulators”, Scientific Research and Essays, Vol. 6, No. 13, pp. 2784-2794, 2011.

[2] Adrian-VasilebDuka, Neural Network Based Inverse Kinematics Solution for Trajectory Tracking of a Robotic Arm, International Conference on Computer Engineering and Applications, Vol. 2, 2011.

[3] Daniel TarnitaA, Dan B. Marghitu, Analysis of a Hand Arm System, Robotics and Computer-Integrated Manufacturing, Vol. 29, pp. 493–501, 2013.

[4] BassamDaya, ShadiKhawandi, Mohamed Akoum, Applying Neural Network Architecture for Inverse Kinematics Problem in Robotics, Journal Software Engineering & Applications, Vol. 3, pp. 230-239, 2010.

[5] Moshan Mirkhani, Rana Forsati, Alireza Mohammed Shari, A Novel Efficient Algorithm for Mobile Robot Localization, International Journal of Robotics and Autonomous, Vol. 61, pp. 920-931, 2013.

[6] Ali T. Hasan1, Hayder M.A.A. Al-Assadi and Ahmad Azlan Mat Isa, Neural Networks’ Based Inverse Kinematics Solution for Serial Robot Manipulators Passing Through Singularities, Artificial Neural Networks-Industrial and Control Engineering Applications, pp. 459-478.

[7] Ramirez A., and A. Rubiano. F., Optimization of Inverse Kinematics of a 3R Robotic Manipulator Using Genetic Algorithms World Academy of Science, Engineering and Technology, Vol. 59, 2011.

[8] Fei Chao a, Zhengshuai Wang a, Changjing Shang b, A Developmental Approach to Robotic Pointing via Human–RobotInteraction, 2013.

[9] F.Y.C. Albert, S.P. Koh, C.P. Chen, S.K.Tiong, S.Y.S. Edwin, Optimizing Joint Angles of Robotic Manipulator Using Genetic Algorithm, International Conference on Computer Engineering and Applications, Vol. 2, 2011.

[10] R. Koker, C. Oz, T. Çakar, and H. Ekiz, A Study of Neural Network Based Inverse Kinematics Solution for a Three-Joint Robot, Robotics and Autonomous Systems, Vol. 49, pp. 227–234, 2004.

[11] A.A.Ata, T.R.Myo, Optimal PointtoPoint Trajectory Tracking of Redundant Manipulators Using Generalized Pattern Search, International Journal of Advanced Robotic Systems, Vol. 2, No. 3, pp. 239-244, 2005.

[12] L. Yingwei, N. Sundararajan, P. Saratchandran’ Identification of Time-Varying Nonlinear Systems Using Minimal Radial Basis Function Neural Networks’IEE Proceedings-Control Theory and Applications, Vol. 44, No. 2, pp. 202-208.

[13] Yingwei, L., Sundararajan, N., and Saratchandran, P.: “A Sequential Learning Scheme for Function Approximation Using Minimal Radial Basis Function Neural Networks’.Technical Report CSP/9505, CenterFor Signal Processing, School of Electrical and Electronic Engineering, Nanyang Technological University, November 1995.