Inverse Kinematics Solution of a Five Joint Robot Using TD and Distributed TDN Network

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 ideas considered 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.

Keywords: Degrees of Freedom, Inverse Kinematics, Time Delay and Distributed TD Neural Network

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).
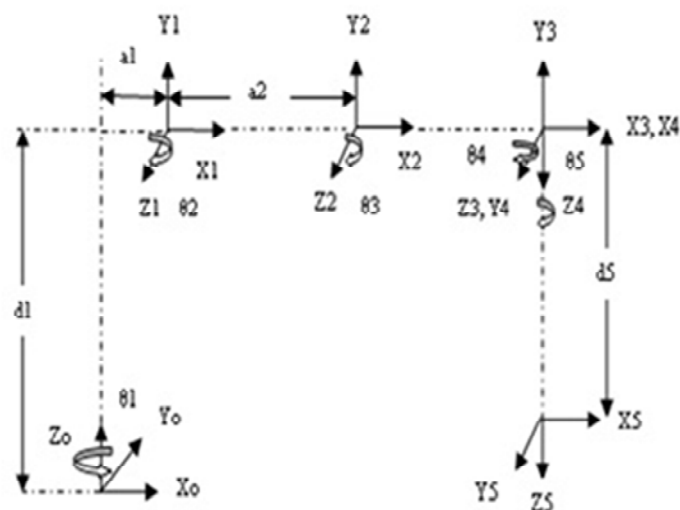

Figure1: Five Joint Manipulator Arm Links and Joints

*   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*
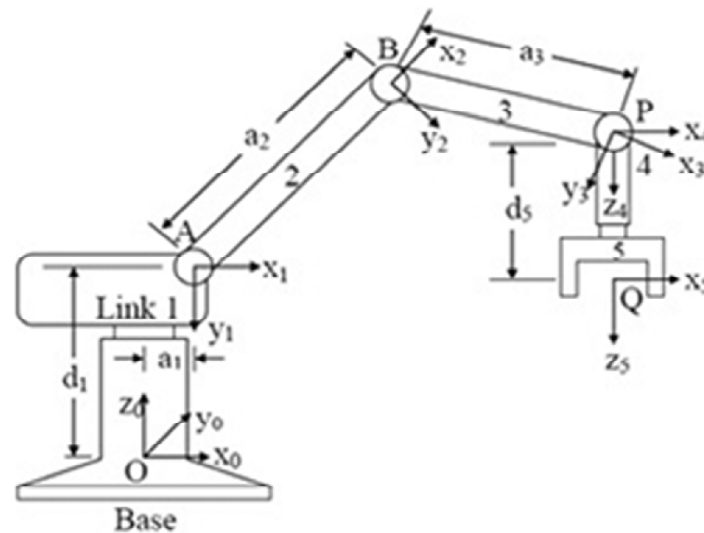
Figure 2: A Five Joint Robot Arm

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 NovelApproach 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 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 SearchAlgorithm (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].

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].

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.

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	$a_3 = 20$	$d_3 = 0$	$\theta_3 = 60°$	$\pm130°$
4	$-\pi/2$	$a_4 = 0$	$d_4 = 0$	$\theta_4 = 50°$	$\pm130°$
5	0	$a_5 = 0$	$d_5 = 5$	$\theta_5 = 70°$	$\pm570°$

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 nth 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 4×4 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 $k = 1$.

- 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:

$$
{}^{i-1}T_i = \begin{bmatrix} C\theta i & -S\theta i\, C\alpha i & S\theta i\, S\alpha i & ai C\theta i \\ S\theta i & C\theta i\, C\alpha i & -C\theta i\, S\alpha i & ai S\theta i \\ 0 & S\alpha i & C\alpha i & di \\ 0 & 0 & 0 & 1 \end{bmatrix}
\tag{1}
$$

Where $S\theta_i = \mathrm{Sin}\,\theta_i$, $C\theta_i = \mathrm{Cos}\,\theta_i$, $S\alpha_i = \mathrm{Sin}\,\alpha_i$, $C\alpha_i = \mathrm{Cos}\,\alpha_i$. The Overall Transformation Matrix,

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

$$
{}^{0}T_1 = \begin{bmatrix} C_1 & 0 & -S_1 & a_1 C_1 \\ S_1 & 0 & C_1 & a_1 S_1 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & \dfrac{1}{\pi} \end{bmatrix}
$$

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

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

$$
{}^{1}T_2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2 C_2 \\ S_2 & C_2 & 0 & a_2 S_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

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

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

$$
{}^{2}T_3 = \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

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

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

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

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

$$
^3T_4 = \begin{bmatrix} 0.6428 & 0 & -0.7660 & 0 \\ 0.7660 & 0 & 0.6428 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}
$$

$$
^4T_5 = \begin{bmatrix} C_5 & -S_5 & 0 & 0 \\ S_5 & C_5 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

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

$$
^4T_5 = \begin{bmatrix} 0.3420 & -0.9397 & 0 & 0 \\ 0.9397 & 0.3420 & 0 & 0 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}
$$

We know that,

$$
^0T_2 = {^0T_1}{^{*1}T_2} = \begin{bmatrix} 0.6123 & -0.6123 & -0.5 & 17.8453 \\ 0.3536 & -0.3536 & 0.866 & 10.3033 \\ -0.7071 & -0.7071 & 0 & -5.6066 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}
$$

$$
^0T_3 = {^0T_2}{^{*2}T_3} = \begin{bmatrix} -0.2241 & -0.8364 & -0.5 & 13.3630 \\ -0.1294 & -0.4830 & 0.866 & 7.7148 \\ -0.9659 & 0.2587 & 0 & -24.9249 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{14}
$$

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

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

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
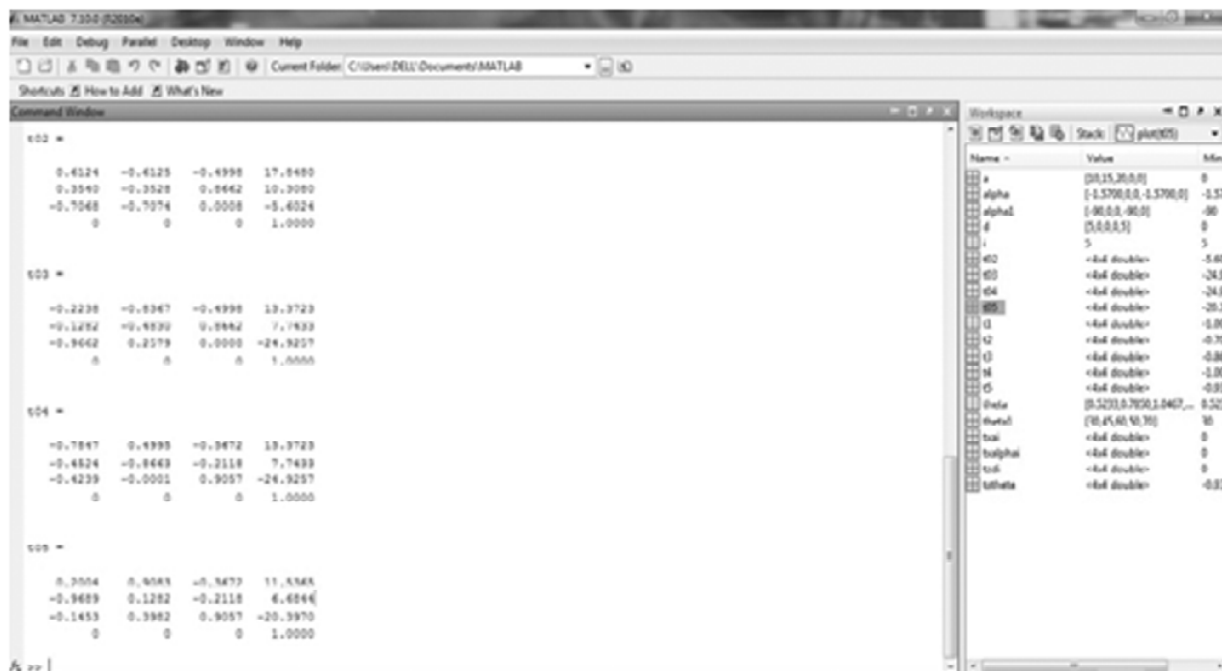

Figure 3: MATLAB Result for the Forward Kinematic

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)

$$ {}^0T_5 = \begin{bmatrix} 0.2015 & 0.9084 & -0.3660 & 11.5330 \\ -0.9688 & 0.1297 & -0.2114 & 6.6578 \\ -0.1446 & 0.3972 & 0.9062 & -20.3939 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix} $$

Case-study: (MATLAB Program Output)

From Table 3.1 Joint Parameter details,

The final matrix ${}_0T^5$ result:

$$ {}^0T_5 = \begin{bmatrix} 0.2004 & 0.9083 & -0.3672 & 11.5365 \\ -0.9689 & 0.1282 & -0.2118 & 6.6844 \\ -0.1453 & 0.3982 & 0.9057 & -20.3970 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \end{bmatrix} $$

The Final values of matrix $({}_0T^5)$ 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:

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

Position Values	Analytical ${}_0T^5$ Values (mm)	Matlab ${}_0T^5$ Values (mm)	Difference (mm)	Analytical ${}_0T^5$ Values (mm)	Robo cell ${}_0T^5$ 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

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

(17)

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

(18)

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

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

(19)

Applying this to yields the following result,

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

(20)

Therefore,

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

(21)

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

$$d_2=-\sin\left(\theta_1\right)p_x+\cos\left(\theta_1\right)p_y$$

(22)

Letting,

$$p_x=\rho\cos$$

(23)

$$p_y=\rho\sin$$

(24)

Where,

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

(25)

$\phi=\text{Arctan } 2\ (p_{y,}\ p_x)$

(26)

We know that,

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

(27)

Therefore,

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

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

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

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 (3.32) and Equation (3.33) 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 θ_3,

$$\theta_3 = \pm\arccos\left(\frac{\left(C_1 p_x + S_1 p_y - a_1\right)^2 + \left(d_1 - p_z\right)^2 - a_2^2 - a_3^2}{2a_2 a_3}\right) \tag{3.37}$$

Next by transferring the matrix $_2^1T$ onto the other side, next equation can be found that will permit us to find θ_2.

$$\left[_2^1T\right]^{-1}\left[_1^0T\right]^{-1}[P] = \left[_4^2T\right] \tag{38}$$

$$\left[_2^1T\right]^{-1} = \begin{bmatrix} C_2 & 0 & -S_2 & -a_1C_2 \\ -S_2 & 0 & -C_2 & a_2S_2 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{39}$$

$$\left[_2^1T\right]^{-1}\left[_1^0T\right]^{-1} = \begin{bmatrix} C_2 & 0 & -S_2 & -a_1C_2 \\ -S_2 & 0 & -C_2 & a_2S_2 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C_1 & S_1 & 0 & 0 \\ -S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & -d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

$$\left[{}_{2}^{1}T\right]^{-1}\left[{}_{1}^{0}T\right]^{-1} = \begin{bmatrix} C_1 C_2 & S_1 C_2 & -S_2 & S_2 d_1 - a_1 C_2 \\ -C_1 S_2 & -S_1 S_2 & -C_2 & C_2 d_1 + a_1 S_2 \\ -S_1 & C_1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{41}$$

$$\begin{bmatrix} C_1 C_2 & S_1 C_2 & -S_2 & S_2 d_1 - a_1 C_2 \\ -C_1 S_2 & -S_1 S_2 & -C_2 & C_2 d_1 + a_1 S_2 \\ -S_1 & C_1 & 0 & -d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} C_3 & -S_3 & 0 & a_3 C_3 + a_2 \\ S_3 & C_3 & 0 & a_3 S_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{42}$$

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*.

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

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

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

$$\theta_4 = 90 - \theta_2 - \theta_3 \qquad (56)$$

Although in this case θ_5 was never changed, by substituting all the values of Table 3.1, in the above formulae, the θ 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 Robocellcould 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'simitate the client interface and menus of Robocell. Scorbaseoperations, menus and commands are depicted in the SCORBASE hand book.

  vi. Robocellgives 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.
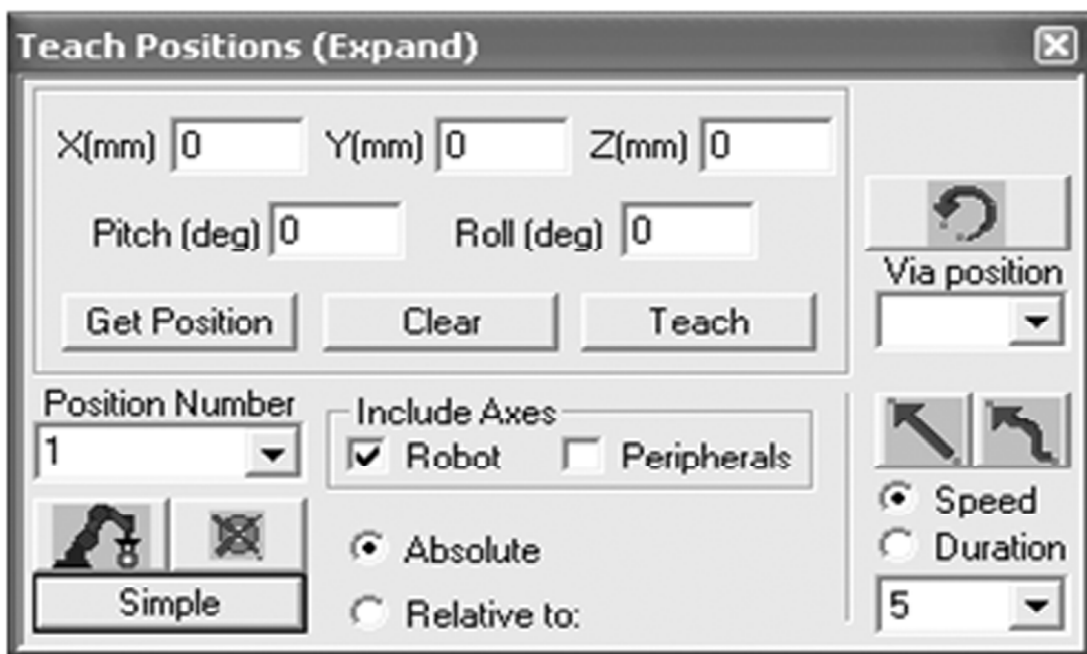

Figure 4: Teach Position (Expand)

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.

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 Robocellis 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.

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

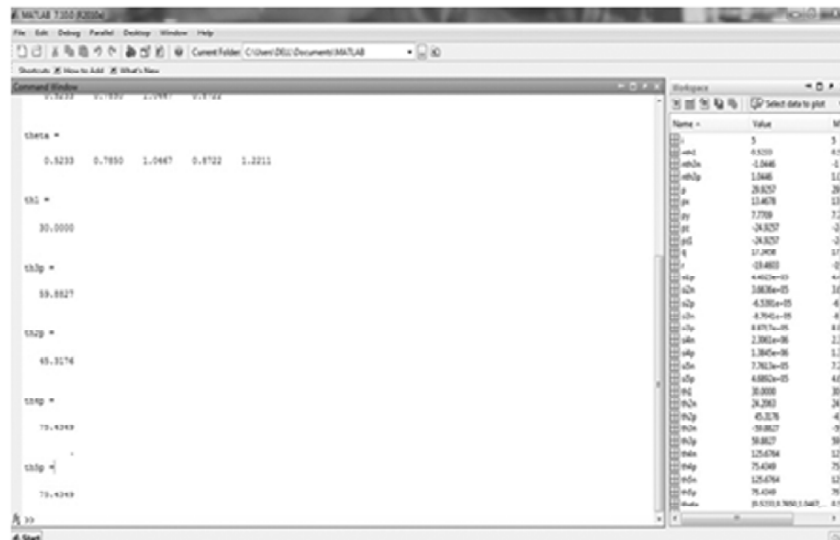| Position Values	Analytical θ Values (°)	Matlab θ Values (°)	Error	Analytical θ Values (°)	Robo cell θ Values (°)	Error
θ_1	30°	30°	0°	30°	30.16°	0.16°
θ_2	45°	45.31°	0.31°	45°	45.27°	0.27°
θ_3	60°	59.88°	0.12°	60°	60.25°	0.25°
θ_4	75°	75.43°	0.43°	75°	75.11°	0.11°
θ_5	75°	75.43°	0.43°	75°	75.11°	0.11°

Figure 5: MATLAB Result for the Inverse Kinematic
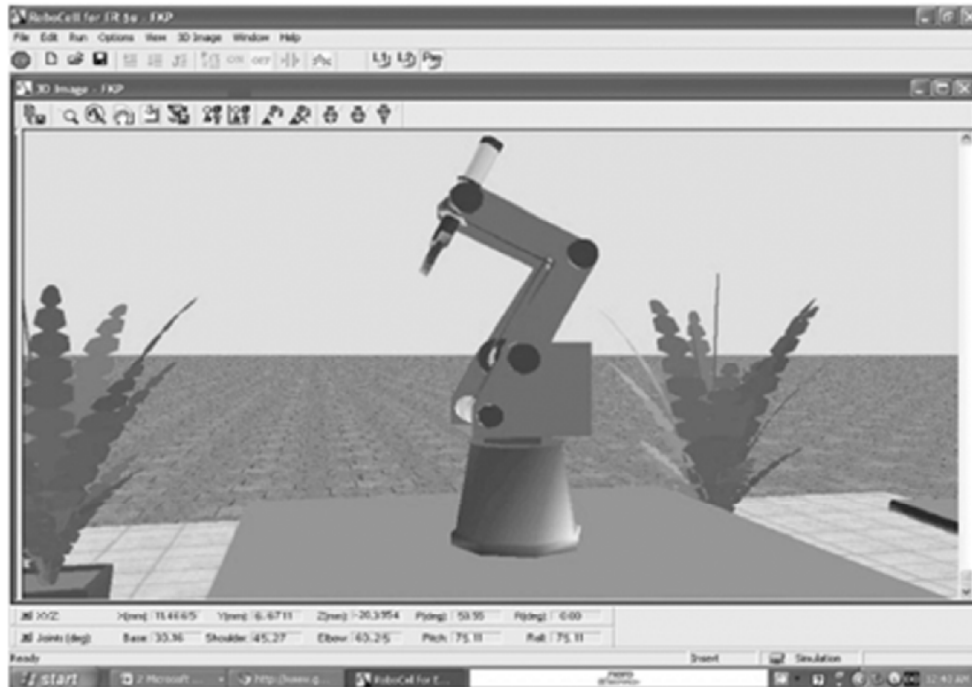
Figure 6:RobocellResult for the Inverse Kinematic

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.

3.2. Time Delay and Distributed Time Delay Neural Network

One special type of multi-layer perceptron structure, the Time Delay NNsperforms by a suitable architecture Figure 7 forthe purpose of extraction of temporal invariant features. This can be achieved by every second layers units which done the feature extraction and estimates the present input as well as the previous inputs at time $t–1$, $t–2$, —— $t–n$ and by this approach a temporal-spatial conversion can be realized.

The training multi-layer networks cannot be used by theLeast Mean Square (LMS) algorithmbecause of the sufficient responses of the nodes in the second layer are not known. A general form of the LMS algorithm called back propagation method which expects a minimum the MSE between the actual output and the desired target.

Many local minima will exhibited in this method rather than the single layer network which has only one minimum. For a node the inputs come from nodes in the previous layer, thus the network is known as feed forward network. The operations like the summation and a sigmoid logistic functionincludes in a node. For each iteration of the minimization process the MSE is decreased up to the error reaches the zero. In the back propagation algorithmthe connection weights modificationscan take place in parallel.

Training in this kind of network is performed by a procedure similar to back propagation that takes the special semantics of coupled links into account. Make the network able to achieve the desired behaviour, the input layer is provided with no of input patterns whose feature shifted within the patterns.
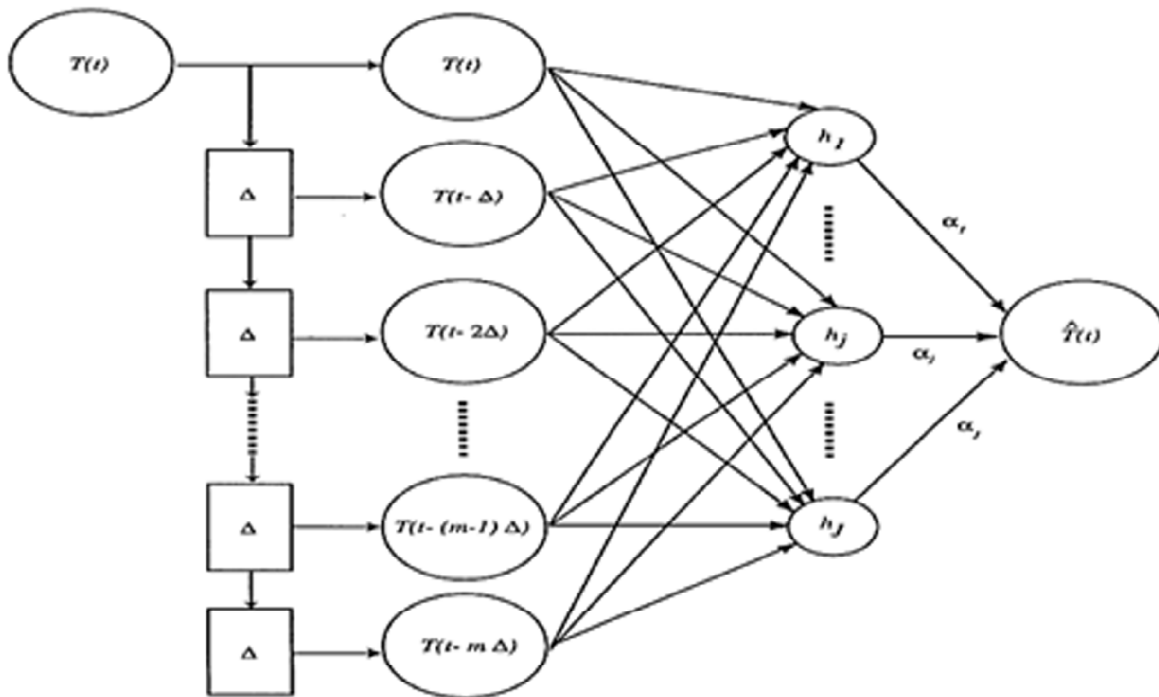
Figure 7: Time Delay NN Architecture

3.3. Time Delay Neural Network (TDNN)

Is anartificial NNarchitecture whose major purpose is to work on sequential data. The TDNN units recognise features independent of time-shift and usually become a part of a huge pattern recognition system. This idea mainly implements in the extraction and evaluation of features which are not dependent on the position.An input signal is improved with delayed copies as other inputs, the NN becomes the time-shift invariant since it has no internal state.

Feature: The part of input that undergo the learning process.

Feature Unit: Those units which are linked with the input pattern.

Delay: Theprevious activations and certain values of the patterns are stored which later helps to recognize patterns positions. Before updating the actual unitsthis can be doneby making a copy of the feature units at each time period. The total number of time steps saved by this method is calleddelay.

Receptive Field: All the provided input is not extracted by the hidden units. They grab the input only from a particular portion which is known as receptive field. Compare to feature units the receptive field is not that much wide and the feature units split up between several receptive fields. Receptive fields must cover all the feature units.

Total Delay Length: This denotes the total length of the layer. It can be calculated by the sum of all delay lengths of the network topology.

Coupled Links: For each consecutive time period every links will be reduplicatedeach link in a receptive field is reduplicated.

The activation of a network is estimated by forwarding the weighted sum to the activation function. Commonly considered activation functions are threshold or sigmoid function.This particular idea is implemented through the introduction of delays in the case of TDNNs.In a NN model, consider a network of n neurons modelled by the equations.

$$x\,i(t) = fi\big(xi(t)\big) + \sum_{j=1}^{n} fij\big(xi(t)\big),\ xj(t),\ i = 1,\ ...,\ n.$$

The variable *xi* represents all the variables describing the physical state of the cell body of the ith neuron in the network. The variable xi represents all the variables describing the physical state of the cell body. If the jth neuron is connected to the ith via a chemical synapse, then the coupling function usually represented as:

$$f_{ij}(x_i(t),\ x_j(t)) = c_{ij}g_{ij}\,(x_j(t))h_{ij}\,(x_i(t))$$

g_{ij} –   a sigmoidal function, usually chosen to have maximum value 1

c_{ij} –   the maximum coupling strength (synaptic conductance)

h_{ij} –   a linear function.

By the same procedure similar to back propagation, the training in this network is also performed by considering more importance to the coupled links as shown in Figure 8. Because of each feature units are reproduced for every time shift, the entire activation data need only at once. The weights of the corresponding connections between the time shifted copies are treated as one because the time varying data are simply the replicates of that event itself. After computing the output error, the error factors are estimated and redirected to input side. As a result the different correction values for the particular network are obtained. Then the next step is averaging the corresponding correction values and the weights are updated with this value.

This update algorithm forces the network to train on time/position independent detection of sub-patterns. This important feature of TDNNs makes them independent from error-prone pre-processing algorithms for time alignment. The drawback is a rather long, computationally intensive, learning phase.

The main condition is the features should cover the input layer. If the network considered needs variable pattern length, we go for the application of a tool to divide these patterns into actual sized pieces. By using the graphical user interface also the patterns can be generated. Here it is the user's responsibility to provideneeded time delayedinputsfor obtaining a successful training.
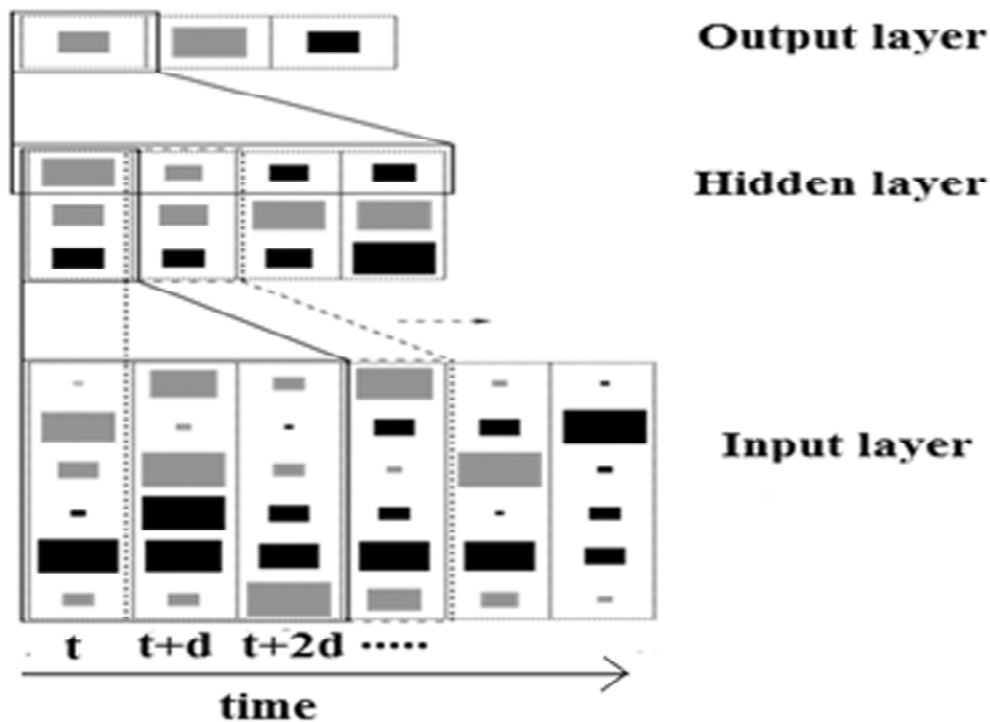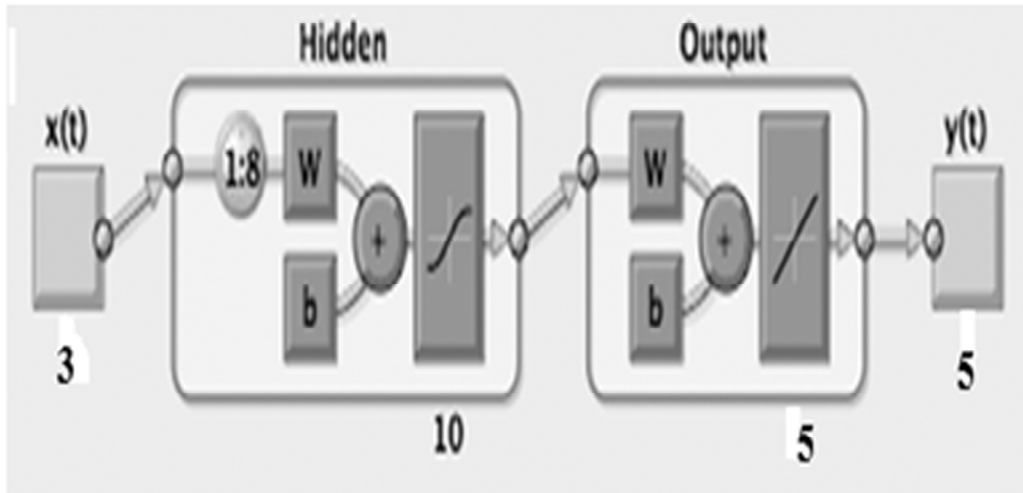

Figure 8: Schematic Diagram of a Time Delay NN
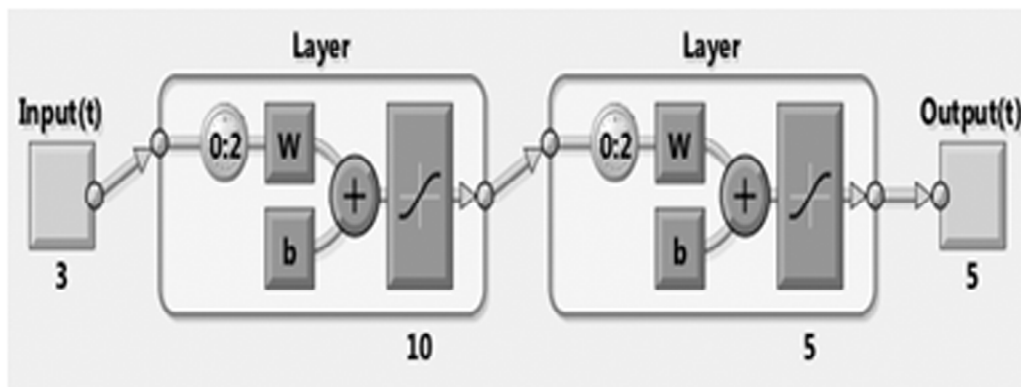
Figure 9: Time Delay Neural Network

Figure 10: Distributed Time Delay Neural Network

The Figure 9 shows the time delay neural network. Three inputs are applied and output obtained as five joint angles and the delay is provided at the hidden layer only.

The Figure 10 below shows a distributed time delay neural network. Here also three layers are there and the delay is distributed throughout the network. The delay 0:2 are provided both in hidden and output layer in the network.

4. RESULT AND DISCUSSION

The positions and orientation corresponding to the Cartesian coordinate system has been analyzed by employing the ANN algorithms such as TDNN and Distributed TDNN 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.

Following is the performance plot Figure 11 obtained by the TDNN and here the best performance obtained as 1.2579-08.The goal is to achieve a zero mean square error and thus the attained output approaches the target.

Next is the depiction of an additional type of TDNN and is Distributed TDNN where the delay is distributed throughout the network. The performance is gained as 3.8343e-10 and it approaches the target value as shown in Figure 12.
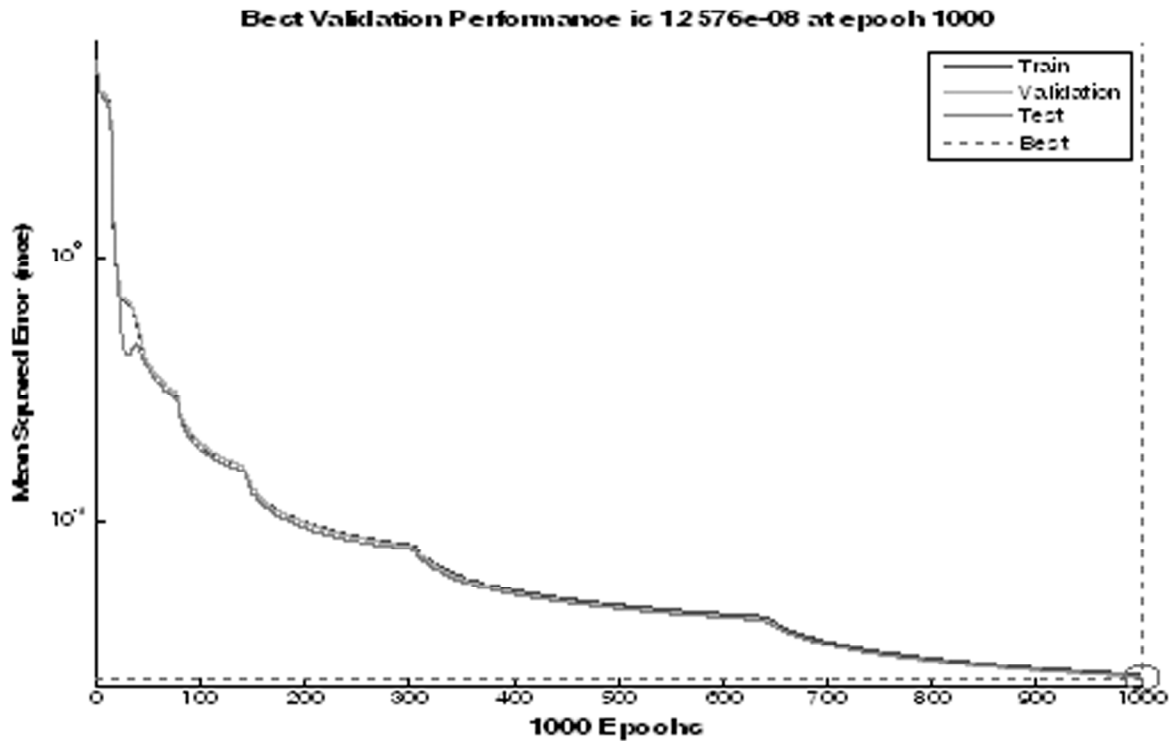
Figure 11: Performance Plot of a TDNN
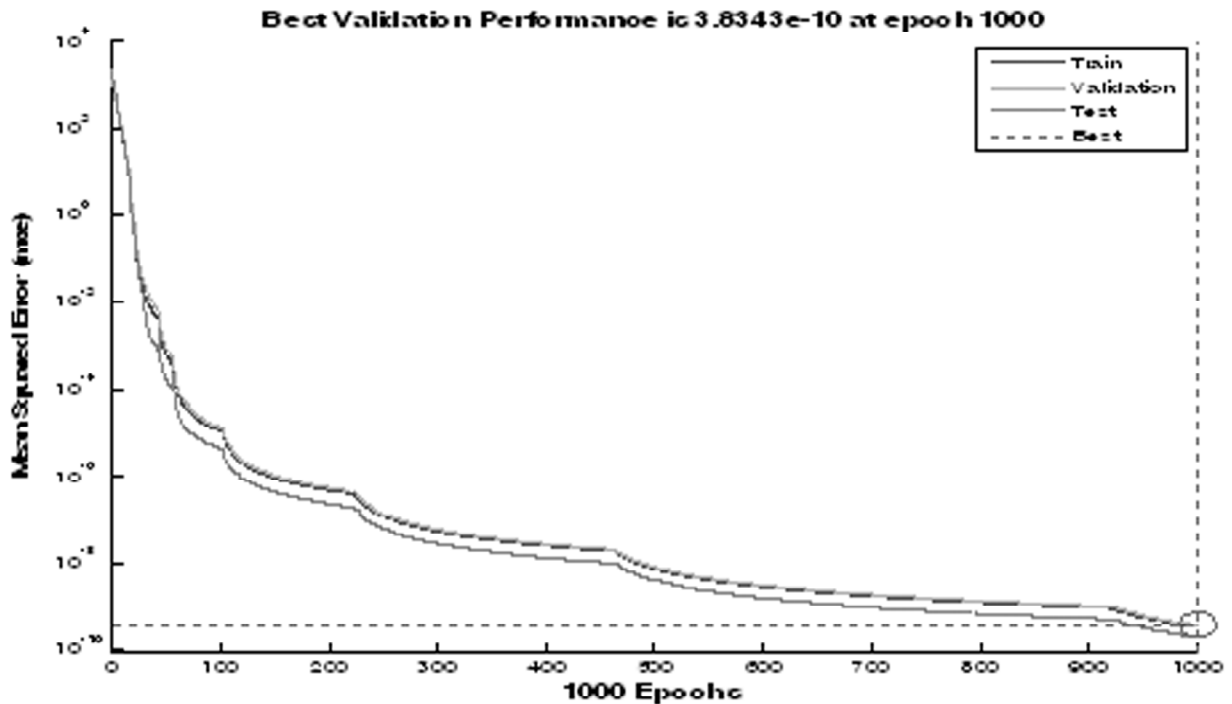
Figure 12: Performance Plot of DTDNN

From the above plot the characteristics of networks can be determined. The best performance is obtained as 9.23416e-28 at epoch 10 and it remains constant. 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 $\theta_1(0, 45)$, $\theta_2(-120, 43)$, $\theta_3(20, 108)$, $\theta_4(-83, 89)$, $\theta_5(0, 70)$ are the joints angles (From Table 4) which are called target angles which means three inputs and five outputs are there and the goal is to obtain zero error which means a high performance algorithm.

The tables shows the desired output which is given as input.The output obtained by the two methods are represented in the following Tables 5 and 6.

Table 4
Desired Taget

	θ_1	θ_2	θ_3	θ_4	θ_5
1.	0	-120.2	95.03	88.81	0
2.	0	-8.93	107.87	-8.93	0
3.	0	-8.88	89.59	9.29	0
4.	0	-2.95	65.09	27.87	0
5.	45	-9.35	105.05	5.70	-45.02
6.	45	-6.17	76.21	19.95	45.02
7.	45	14.78	20.48	54.73	45.02
8.	67.19	-9.63	100.80	-1.17	0
9.	21.56	-4.43	69.91	24.53	0
10.	38.10	-15.86	98.92	-83.06	0
11.	30	42.30	58.48	-10.79	70

Table 5
Output Obtained by TD Algorithm

	θ_1	θ_2	θ_3	θ_4	θ_5
1.	0.0004	-120.27	95.03	88.80	0.0007
2.	0.00009	-8.93	107.87	-8.93	0.0006
3.	0.0004	-8.88	89.59	9.29	0.0001
4.	0.0005	-2.95	65.09	27.87	0.0003
5.	45.00	-9.35	105.05	-5.70	-45.02
6.	45.00	-6.17	76.21	19.95	45.02
7.	45.00	14.78	20.48	54.73	45.02
8.	67.19	-9.63	100.80	-1.17	0.0007
9.	21.56	-4.43	69.91	24.53	0.0003
10.	38.10	-15.86	98.92	-83.06	0.0005
11.	30.00	42.30	58.48	-10.78	69.99

Table 6
Output Obtained by Distributed TD Algorithm

	θ_1	θ_2	θ_3	θ_4	θ_5
1.	0.0003	-120.27	95.03	88.80	0
2.	0.00008	-8.93	107.87	-8.93	0.0002
3.	0.0006	-8.88	89.59	9.29	0.0001
4.	0.0001	-2.95	65.09	27.87	0
5.	45.00	-9.35	105.05	-5.70	-45.02
6.	45.00	-6.17	76.21	19.95	45.02
7.	45.00	14.78	20.48	54.73	45.02
8.	67.19	-9.63	100.80	-1.17	0.0002
9.	21.56	-4.43	69.91	24.53	0.0004
10.	38.10	-15.86	98.92	-83.06	0.0002
11.	30.00	42.30	58.48	-10.79	69.99

Table 7
Comparison of TDNNand Distributed TDNN

Algorithms Used	Time (sec)	Performance(MSE)	Gradient
TDNN	22	1.2576e-08	$6.7801e^{-05}$
Distributed NN	27	3.8343e-10	$6.4595e^{-05}$

By considering and analysing the obtained outputs from TDNN and Distributed NN given in Table 5 and 6 with reference to Table 3 and 4, it can be concluded that the estimated output is very much closer to the desired target. Thus the goal is attained by using both of these algorithms.The Table 7 provides a comparison of the algorithms TDNN and Distributed NN. It evaluated that the time consumption is minimum for TDNN than Distributed TDNN and the performance of Distributed TDNN is achieved as better since it approaches the zero value more than other methods.

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, Artificial Neural Network and can be analyzed by the performances of the adopted neural network algorithms such as TDNN and Distributed NN. 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 (13), pp. 2784-2794, 2011.

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

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

[4] Bassam Daya, Shadi Khawandi, 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. RubianoF. 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, IPCSIT, 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.