EJONS International Journal on Mathematic, Engineering and Natural Sciences | ISSN 2602 - 4136

Article Arrival Date: 24.11.2020 | Article Type: Research Article | Article Published Date: 15.12.2020

Doi Number: http://dx.doi.org/10.38063/ejons.362

ARTIFICIAL NEURAL NETWORK APPLICATION FOR INVERSE KINEMATIC SOLUTION OF A 5-DOF ROBOTIC ARM

Aysun EĞRISÖĞÜT TIRYAKI

Sakarya University, Faculty of Engineering, Department of Mechanical Engineering, Sakarya,Turkey. aysune@sakarya.edu.tr, ORCID: 0000-0001-6440-8396

ABSTRACT

The inverse kinematic solution is one of the major problems for real time application of robot control. The conventional solution methods such as geometric, algebraic and numerical iterative are insufficient and slow in the inverse kinematic solution. Recently, different solution methods based on artificial intelligence techniques have been developed in order to solve inverse kinematics of especially redundant robots and to reduce the solution time. In this study, an Artificial Neural Network (ANN) model was designed to solve the inverse kinematics of the 5 degrees of freedom (DOF) robotic arm. Network was trained using database obtained from the experimental setup and ANN model was validated by experimental works. The Neural Network (NN) calculated the joint angles with high accuracy with respect to the given (x, y, z) Cartesian coordinates. The comparison results show that the proposed NN has a high correlation (R=0.9988) and superior performance for solving inverse kinematics of the robotic arm.

Keywords: Inverse kinematics, Robotic arm, Artificial neural network

1. INTRODUCTION

Robot manipulators are widely used in various industrial automation applications and also the other specialized fields as medical, military. Manipulator applications involve pick and place an object in a specific position, precision grasping and manipulation of the objects. To accomplish these, the end-effector motion which should be efficiently controlled is defined. Robot kinematics defined mapping between joint space and Cartesian space (x, y, z) is needed for the position control of robot manipulators. Forward kinematics and inverse kinematics are used for the kinematic analysis of robot manipulators. Forward kinematics compute the position, orientation and velocity of the end effector from the joint displacements and angles, whereas inverse kinematics compute the joint displacements and angles from the end effectors position and velocity.

The inverse kinematics solution is generally difficult for control of robot manipulators. The inverse kinematic problem is traditionally solved using the geometric, algebraic and iterative models. These have certain disadvantages, for example, the closed-form solutions are not guarantee by algebraic methods, and if the geometric method is used closed-form solutions of the first three joints must be geometrically. A closed form or analytic solution can be determined using geometric and algebraic approaches. But, this solution becomes more challenging as the robot joints increases, and closed-form solution do not exist for some serial-link manipulators. The numerical iterative inverse kinematics solution converges to a single solving based on beginning point. Also, in these conventional solution methods, the advanced complexity in the geometrical structure of the robot can cause in a prohibitive computational cost. Therefore, researchers have studied on different solution methods based on artificial intelligence techniques for solving the inverse kinematics.

Köker (2013) presented the inverse kinematics solution for the six-joint Stanford robot by combining characteristics of neural network and the genetic algorithms. Tejomurtala and Kak (1999) analyzed backpropagation feed-forward neural networks whose weights are described in terms of sine and cosine to match the forward kinematics of the robot. Karlik and Aydin (2000) investigated the best neural-network configurations for solving the inverse kinematics of a six-joint robot manipulator. To achieve the inverse kinematics solutions, placement and orientation angles of robot were used. Köker et al. (2004) proposed the neural network to solve inverse kinematics using cubic trajectory planning for the three joint robot manipulator. Bingul et al. (2005) applied backpropagation neural network to inverse kinematics problem for 6R robot manipulator with an offset wrist which does not exist closed form solution. Mayorga and Sanongboon (2005) presented an neural network for fast calculation of the inverse kinematics and effective geometrically bounded singularities prevention of redundant manipulators. Hasan et al. (2006) presented ANN with a learning algorithm based on adaptive updating of the weights of the network to solve the inverse kinematics problem. 6-DOF manipulator was simulated to control of x-y-z motion. Daya et al. (2010) presented a neural network architecture which consist of 6 sub-neural to control the position of robotic manipulators. Duka (2014) used feed-forward neural network computed desired trajectories in the two-dimensional Cartesian coordinate system for the three-link planar manipulator. Zacharie (2012) implemented Logistic Belief Neural Network (LBNN) on a mobile robot with two gripped arms. LBNN was designed to control five degrees of freedom robot arm in real time. Jiang and Ishita (2008) presented control method consisting of traditional controller and neural network controller for trajectory tracking control of industrial robot manipulators. Neural network controller played the major role in the generating of the actuating force/torque required by the dynamic trajectory. El-Sherbiny et al. (2018) compared ANN, adaptive neuro fuzzy inference system (ANFIS) and genetic algorithm techniques used to solve the inverse kinematics problem of the 5-DOF robot arm. Xu et al. (2019) suggested recurring neural network based controller for redundant manipulator subject to kinematic uncertainties. The controller can provide robustness and adaptability to even under uncertain conditions because it is able to learning uncertain model parameters online.

The control of robotic manipulators has a long history because of difficult manipulation tasks and has a wide range of research areas for researchers. Even though many robots work with high accuracy, repeatability, and stability, research continue in improvement of robot manipulators for the extra enhancement of precision in robot control. In real time control of the robot manipulator, one of the most important and difficult problems to solve is the inverse kinematics solution. In this study, a robotic arm with 5 degrees of freedom that is similar to the human arm was built. Optimum ANN configuration was investigated to solve the inverse kinematics for the purpose of position control of the robotic arm in real time. The designed NN model was trained using the database obtained from the experimental setup. After training completes, validation of the ANN model was carried out using new experimental data that never encountered before.

2. MATERIAL AND METHODS

2.1. The Robotic Arm

Similar to the human arm, the robotic arm which has 5-DOF consists of basic structures which comprises connected to three subparts and a gripper. These subparts are shoulder, upper arm, and forearm. The shoulder is a length of 100 mm and a weight of 282 gr while the upper arm and forearm have 250 mm length, 588 gr and 520 gr weight respectively. The distance from the center of gripper to the wrist is 100 mm and a weight of 172 gr.

Mechanical structure design was carried out using “Sheet Metal Module” in Solidworks (Figure 1). The robotic arm was built from aluminum material due to its lightness using the proper manufacturing process.

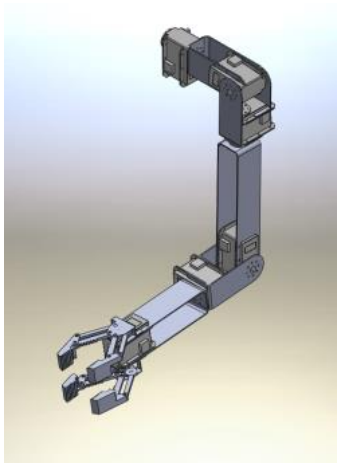


Figure 1. 3D CAD model of the robotic arm

All degrees of freedom are driven by servo motors. Dynamixel MX-28R and MX-64R servo motors given technical specifications in Table-1 were used for rotation of the joints.

Table 1. Technical specifications of the servo motors

	Dynamixel MX-28R	Dynamixel MX-64R
Stall Torque	2.3N.m (11.1V,1.3A) 2.5N.m (12V, 1.4A) 3.1N.m (14.8V,1.7A)	5.5N.m (11.1V,.9A) 6.0N.m (12V, 4.1A) 7.3N.m (14.8V, 5.2A)
Weight	72 g	162 g
No-load Speed	50rpm(11.1V) 55rpm (12V) 67rpm(14.8V)	58rpm (11.1V) 63rpm (12V) 78rpm (14.8V)
Reduction Ratio	193:1	200:1

2.2. Kinematic Analysis of the Robotic Arm

2.2.1 Forward Kinematics

In forward kinematics, the goal is to obtain coordinates of end-effector, given the known joint coordinates. The robotic arm has 5-DOF which are three at the shoulder and two at the elbow. All joints of the arm are revolute. The coordinate frame to each link has been appointed considering the Denavit-Hartenberg (D-H) convention which is represented in Figure 2. The D-H parameters are indicated in Table 2, where θi, α_i, d_i, a_i represent rotation about the Z-axis, rotation about the X-axis, transition along the Z-axis, and transition along the X-axis respectively.

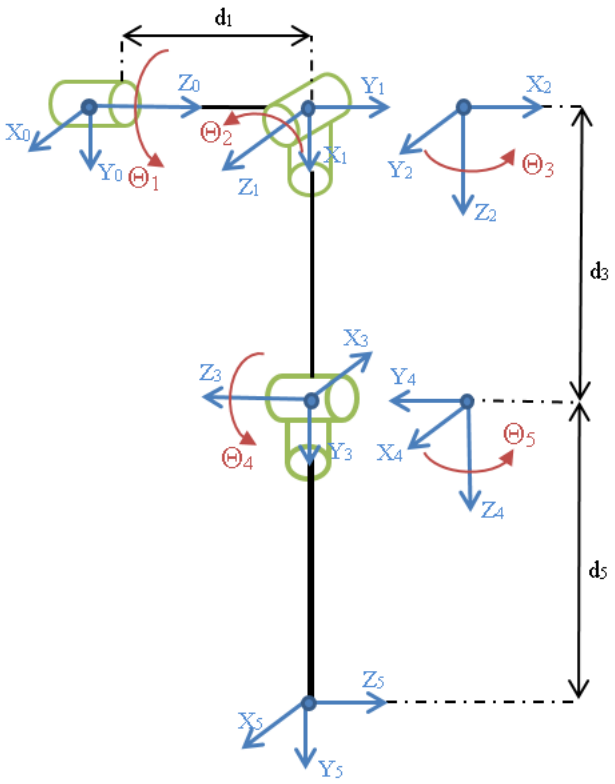


Figure 2. The coordinate frame to each link

Table 2. Denavit-Hartenberg parameters

	d_i	α_i	a_i	θ_i
1	d_1=100	90º	0	θ_1+90º
2	0	90º	0	θ_2+90º
3	d3=250	90º	0	θ_3+90º
4	0	-90º	0	θ_4
5	d_5=250	90º	0	θ_5

According to the Denavit-Hartenberg, the homogeneous transformation matrix for two adjacent frames is given as:

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

If the D-H parameters in Table 2 substituted into (1), the transformation matrices of each adjacent frame are obtained:

$$T_1^0 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} \cos\theta_2 & 0 & \sin\theta_2 & 0 \\ \sin\theta_2 & 0 & -\cos\theta_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} \cos\theta_3 & 0 & \sin\theta_3 & 0 \\ \sin\theta_3 & 0 & -\cos\theta_3 & 0 \\ 0 & 1 & 0 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} \cos\theta_4 & 0 & -\sin\theta_4 & 0 \\ \sin\theta_4 & 0 & \cos\theta_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} \cos\theta_5 & 0 & \sin\theta_5 & 0 \\ \sin\theta_5 & 0 & -\cos\theta_5 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

The transformation matrix T_0^5 which is from base link to end-effector can be obtained as product of matrices for every joint given in 2:

$$T_5^0 = T_1^0 . T_2^1 . T_3^2 . T_4^3 . T_5^4 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

where ***p*** (p_x,p_y,p_z) is position vector of the end-effector and ***n*** (n_x,n_y,n_z), ***o*** (o_x,o_y,o_z) and ***a*** (a_x,a_y,a_z) are orthogonal unit vectors that define orientation of end-effector frame.

2.2.2 Inverse Kinematics

In inverse kinematics, the goal is to compute joint displacements, given the reference orientation and position for the end-effector. In this study, ANN approach were used for solving the inverse kinematics problem.

2.2.2.1 Analytical Inverse Kinematic

As given in (3), the relationship between the reference base frame and the end point is:

$$T_1^0 . T_2^1 . T_3^2 . T_4^3 . T_5^4 = T_{end} \quad (4)$$

If the both side of equation (4) product with $(T_0^1)^{-1}$:

$$\underbrace{T_2^1 . T_3^2 . T_4^3 . T_5^4}_{L_1} = \underbrace{(T_1^0)^{-1}\, T_{end}}_{R_1} \quad (5)$$

Left and right side of equation (5) denote L_1 and R_1 respectively. The equations $L_1(3,2)=R_1(3,2)$ and $L_1(3,4)=R_1(3,4)$ must be satisfied. These relations can be used to solve for θ_1. This yield:

$$\theta_1 = atan2(p_x - o_x\, d_5, -o_y\, d_5 + p_y) \quad (6)$$

The equations $L_1(2,2)=R_1(2,2)$ and $L_1(2,4)=R_1(2,4)$ can be also derived and these relations provide the solution for θ_2:

$$\theta_2 = acos(\ (o_z\ d_5 + d_1 - p_z)/d_3) \quad (7)$$

Equation (8) can be obtained to product with $(T_2^1)^{-1}$ both side of equation (5):

$$\underbrace{T_3^2.T_4^3.T_5^4}_{L_2} = \underbrace{(T_2^1)^{-1}\ (T_1^0)^{-1}.T_{end}}_{R_2} \quad (8)$$

Left and right side of equation (8) denote L_2 and R_2 respectively. Using the relation $L_2(3,4)=R_2(3,4)$, $L_2(3,2)=R_2(3,2)$ and $L_2(3,4)=R_2(3,4)$, θ_4 and θ_3 can be provided respectively as follows:

$$\theta_4 = \text{acos}(((d_1 - p_z)\cos(\theta_2) + (p_x\ cos(\theta_1) + p_y\ sin(\theta_1))\sin(\theta_2) - d_3)/d_5) \quad (9)$$

$$\theta_3 = atan2(o_x\cos(\theta_1)\cos(\theta_2) + o_y\cos(\theta_2)\sin(\theta_1) + o_z\sin(\theta_2)\,, o_y\cos(\theta_1) - o_x\sin(\theta_1)) \quad (10)$$

The equations $L_2(3,1)=R_2(3,1)$ and $L_2(3,3)=R_2(3,3)$ can be used to solve θ_5. These yields:

$$\theta_5 = atan2(-a_z\cos(\theta_2) + a_x\cos(\theta_1)\ \sin(\theta_2) + a_y\sin(\theta_1)\sin(\theta_2)\,, -n_z\cos(\theta_2) + a_z\cos(\theta_1)\ \sin(\theta_2) + n_z\sin(\theta_1)\sin(\theta_2)) \quad (11)$$

2.2.2.2 The Artificial Neural Network Approach

ANN has been applied to different research in many fields of science and successfully solve complex and nonlinear problems, thanks to its ability to learn and generalize. Neural networks are basically sets of mathematical functions modeled in a computer program, which simulate basic abilities of human brain and neural biology.

An input layer, at least one hidden layer, and one output layer exist in the multilayer feed-forward neural network. Each layer contains processing elements called neurons. Neurons in layers are joined by connection weights associated with real numbers. The weights are updated in the training operation of the neural network. The sum input (x_i) to layer neuron (i), is sum of the weight (w_{ij}), multiplied by the input value (x_j) from the previous layer neuron (j) for each connection.

$$x_i = w_{i0} + \sum_{j=1}^{N} w_{ij} \times x_j \quad (12)$$

where N is the number of inputs, w_{i0} is the bias of neuron. The bias is generally considered equal to 1. Therefore, the corresponding weight could shift the activation function throughout the abscissa axis (Haykin, 1999).

Output of the i neuron, Vi is as follows:

$$V_i = f(x_i)$$

where f is the activation function. In the neural network training process, Q set training data are offered to network. An iterative algorithm regulates the weights because the outputs (y_k) obtained with respect to the inputs should be as close as possible to the target outputs (d_k). For neural network with total output number of K, the Mean Square Error (MSE) function that should be minimized is as follows;

$$MSE = \frac{1}{Q \times K} \times \sum_{q=1}^{Q}\sum_{k=1}^{K}\left[d_k(q) - y_k(q)\right]^2 \quad (13)$$

In this study, back propagation algorithm is used to minimize MSE by updated the weights of connections. Updated weights are calculated as follows;

$$w_{ij}^{t+1} = w_{ij}^{t} + \Delta w_{ij}^{t+1} \quad (14)$$

where Δw_{ij}^{t+1} is the (+/-) incremental change in the weight and this is assigned by the Levenberg-Marquardt optimization as follows;

$$\Delta w_{ij}^{t+1} = \left[J^T J + \mu I\right]^{-1} J^T e = \left[H + \mu I\right]^{-1} g \quad (15)$$

where I is the identity matrix, μ is adaptive training parameter, J is Jacobean matrix and e is all errors.

The adjusted weights and bias are given sweep rearwards by neural network. Thus, new outputs of the neural network are computed. The MSE is recomputed using $w_{ij}^{t} + \Delta w_{ij}^{t+1}$. If this new MSE value is less than calculated in the previous, training parameter is decreased μ by μ^{-}. If the MSE value is not reduce, it is increased μ by μ^{+}. The algorithm is supposed to have converged when the gradient norm is smaller than some preconcert value, or when the MSE has been decreased to some error aim.

When weights reach ideal values that allow the network to produce outputs that are close enough to desired outputs, the training process of ANN is completed. After the weights are adjusted, the ANN model is tested by giving only varied input data. If the ANN model replies correctly to input data not included in the training data, it is said that it has generalization ability.

2.3. Experimental Setup

In this paper real time applications of the robotic arm were performed. The fundamental configuration of robotic arm is shown in Figure 3.

Figure 3. Fundamental configuration of robotic arm

The experimental setup has two parts, hardware and software. Hardware consist of power supply, computer, usb converter and servo motors. MX series Dynamixel DC servo motors were used to set up the robotic arm. According to the stall Torque, MX − 28R, MX − 64R are assigned to form the wrist, elbow and shoulder respectively. Computer comprises i7 cpu, 4 GB Ram and onboard graphic card. All motors contain a microcontroller with RS-485 interface. The RS-485 protocol is employed for communication between computer and the arm. Servo motors are connected with computer through “USB2Dynamixel”. USB2Dynamixel, is a device to convert a signal from R485 to USB between computer and servo motors.

Algorithm was developed using Matlab-Simulink software. Communication between Matlab-Simulink and servo motors are established using RapidSTM32 library. RapidSTM32 is a Simulink device driver block set and tool suite for STM32 microcontroller family.

3. NEURAL NETWORK IN IMPLEMENTATION FOR THE INVERSE KINEMATIC

The artificial neural network was applied to solve the inverse kinematic of 5-DOF robot arm in this section. Neural network model was developed based on the experimental work. Training data and test data was generated using database obtained from the experimental setup. Neural network model was trained using 480 randomly selected data points and 113 data points were utilized to test network performance and validate of the predictive capability of the model. Cartesian coordinates (x, y, z) were chosen as input variables and joint angles (θ_1, θ_2, θ_3, θ_4, θ_5) were set as the output variables. Figure 4 depicts the schematic structure of the neural network used in the study, with its three inputs and five outputs. The different combination of Cartesian coordinates (x, y, z) used as training input data is shown in Figure 5.

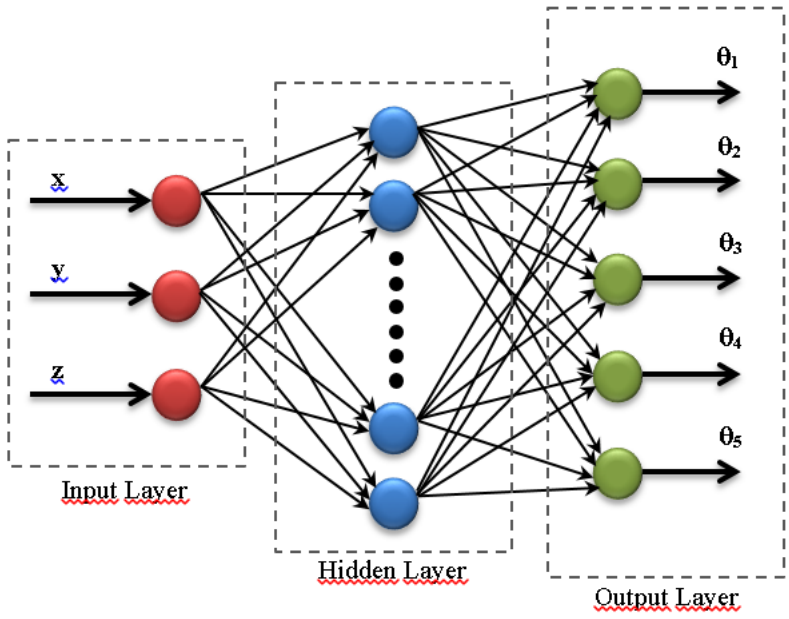


Figure 4. Configuration of multilayer neural network for solving the inverse kinematic

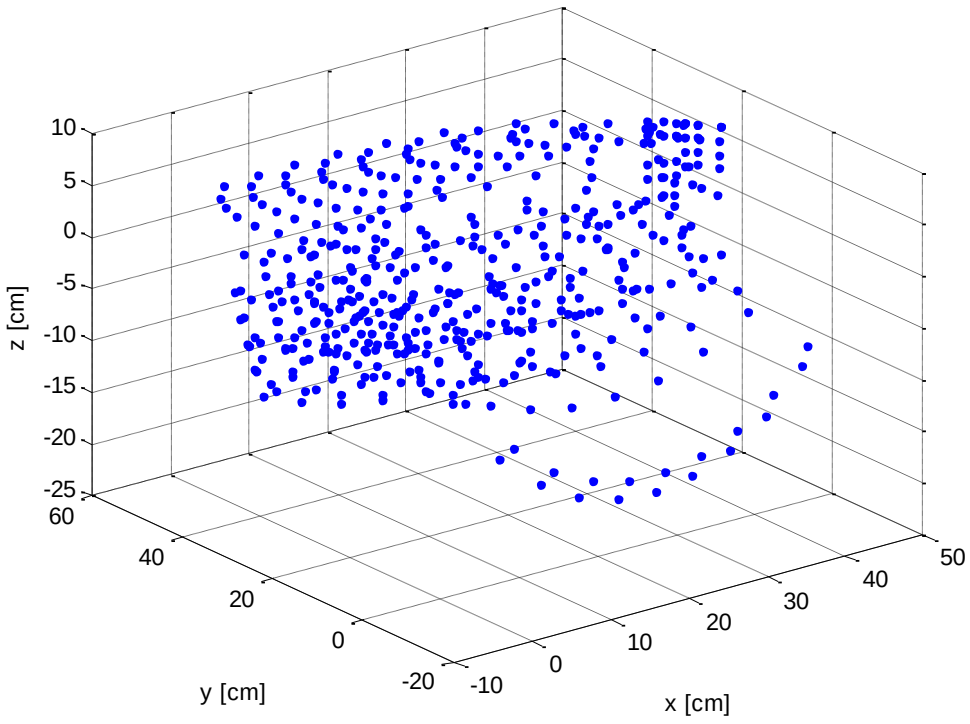


Figure 5. Training input data

Feed-forward neural networks which have varied topologies (number of layers, number of neurons), various training algorithm and parameters and activation functions were designed and the optimum network configuration was investigated. Each network was trained to different levels and, among all, the one qualified by the global minimum generalization error and the fastest was selected. Table 3 shows functions and architecture of the neural network proposed for inverse kinematic solution. The optimum ANN model configuration was obtained to be 3-10-5 with 10 neurons in hidden layer.

Table 3. Functions and architecture of the neural network.

Network	Feed-forward backpropagation network
Training method	Supervised training
Transfer function	Hyperbolic tangent sigmoid function (Hidden Layer) Pure linear function (Output Layer)
Training function	Levenberg-Marquardt
Learning function	Gradient descent
Performance function	Mean squared error

The trial MSE values against the iteration number using selected ANN topology was indicated in Figure 6. The ANN was trained with a learning rate of 0.2 and momentum coefficient of 0.5 at 8816 epochs. It can be seen from Figure 6 that the values of MSE converged to approximately 1,74×10-3 in 8816 epochs and considering structure of the neural network model, training of the network was determined to be successful.

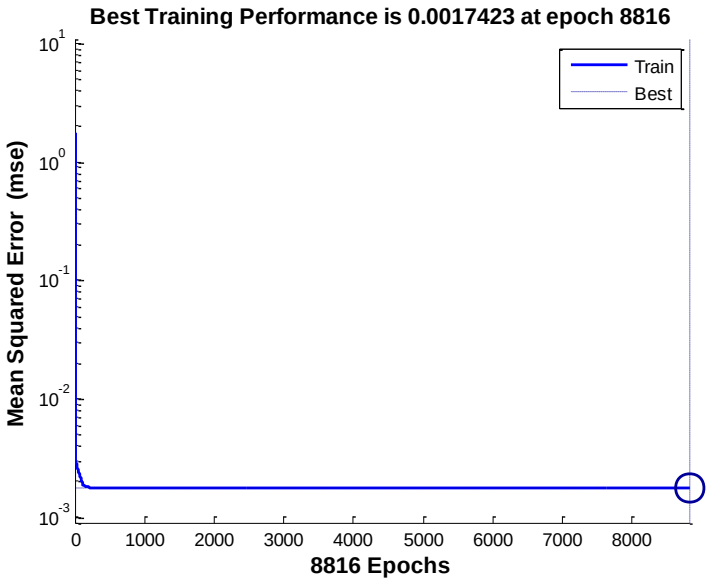


Figure 6. Training performance of the neural network

4. RESULTS AND DISCUSSION

When the values obtained from the experiments and the prediction results of the ANN model were compared, it was seen that there was a strong correlation between them. The correlation between

model output and the desired values to represent the training performance of optimal ANN topology is shown in Figure 7. Coefficient of determination (R) of 0.9988 was obtained for predicting joint angles. This shows that the ability of ANN to solve the inverse kinematic was very good.

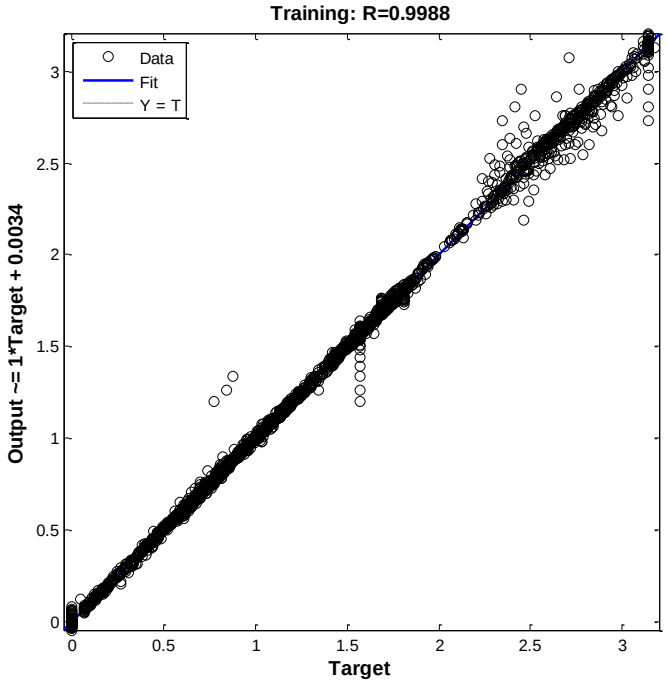


Figure 7. The correlation between ANN model output and the target values

After training, validation of the ANN model was established giving fully unseen data (i.e. test data). The different combination of Cartesian coordinates (x, y, z) given as input data to NN model is shown in Figure 8.

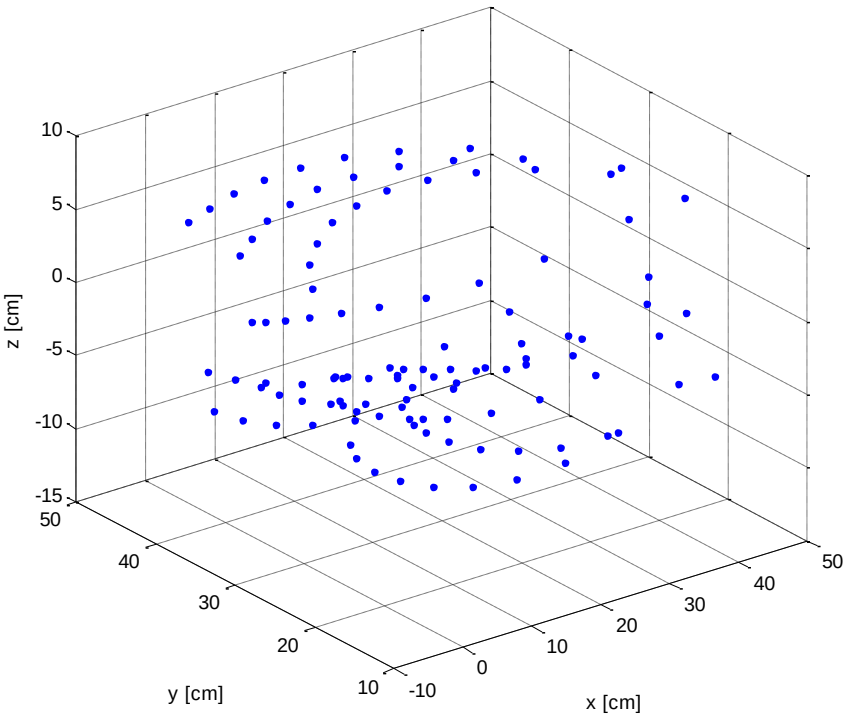
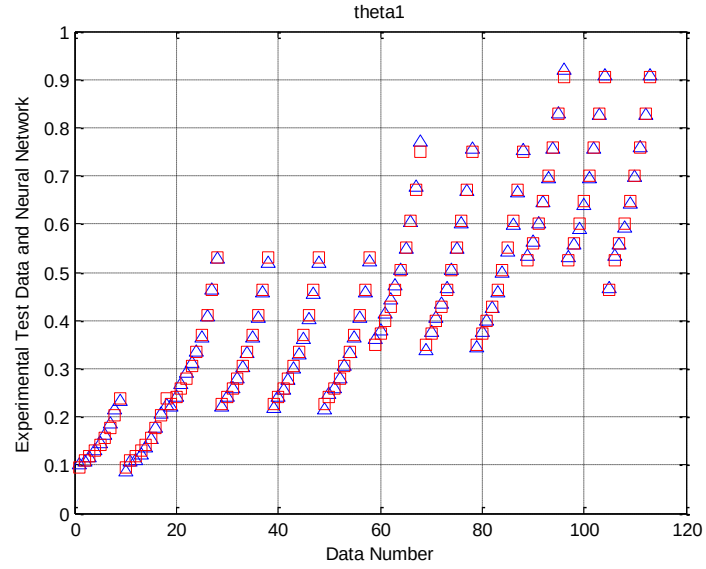
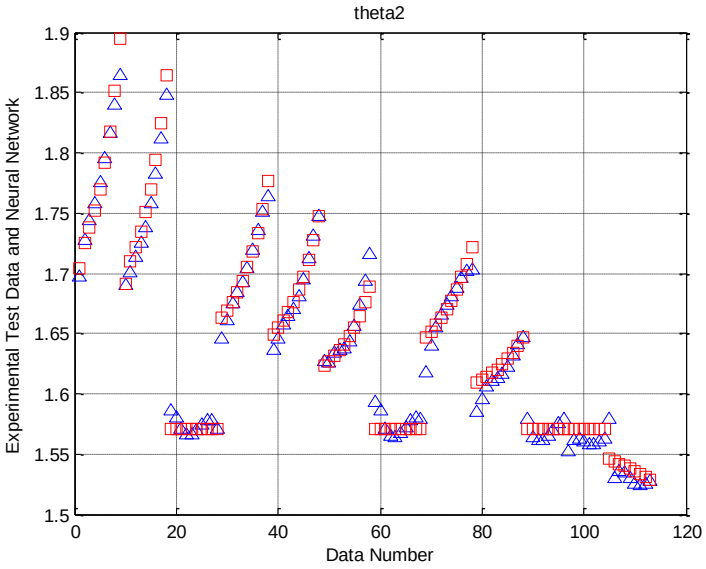
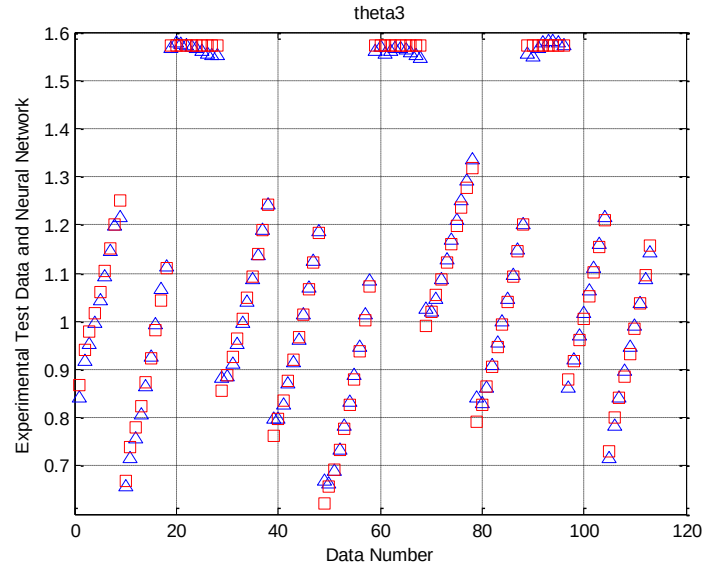


Figure 8. Validation data set

The comparative parity plot for validation results of ANN model and experimental joint angles (θ_1, θ_2, θ_3, θ_4, θ_5) is indicates in Figure 9. As can be seen in test (validation) results, neural network model has a very strong prediction and a good generalization on solving inverse kinematic problem.







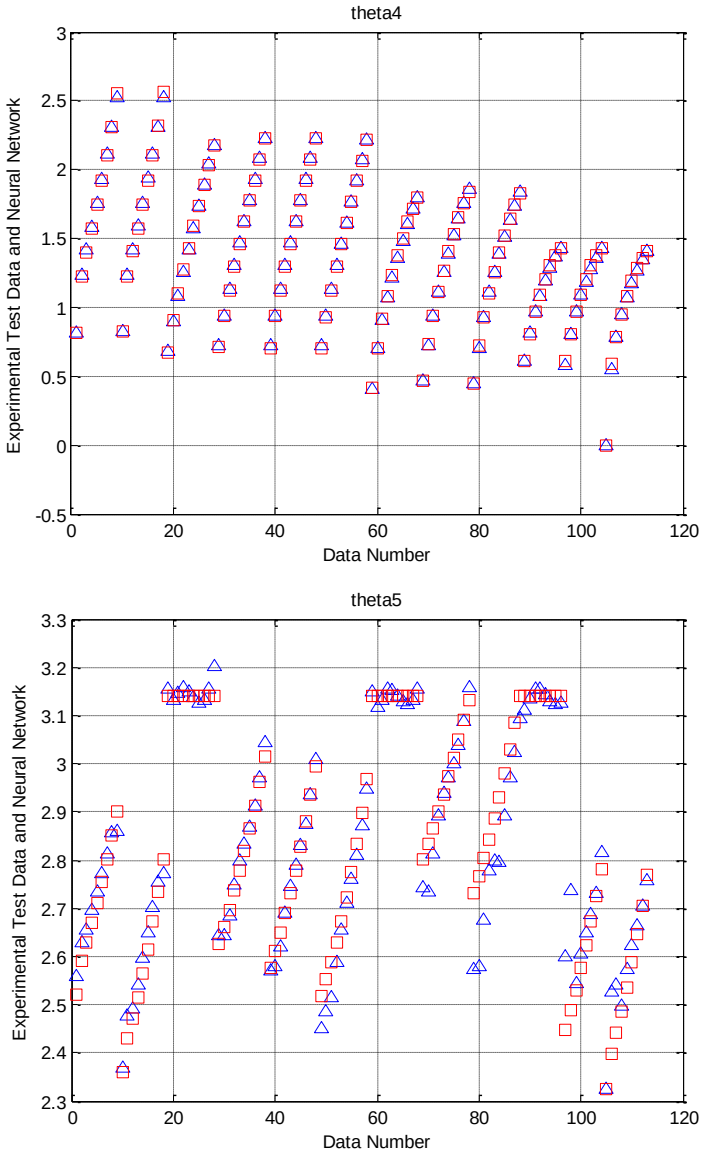


Figure 9. Experimental values and ANN predictions of joint angels for validation data set

5. CONCLUSION

In this study, the NN model was created to solve the inverse kinematics problem and predict joint angles for a 5-DOF robotic arm. Then NN was trained until the error is acceptable using experimental data. The training performance of the designed network was evaluated and a high correlation (R=0.9988) was obtained. Considering the validation results, it is seen that the NN method has good absolute fit to the reference. As a result, NN model has solved inverse kinematic problem with high accuracy, has provided a strong estimate and good generalization capability. Therefore, this network model can be used to compute joint displacements needed to move the end effector to a desired position.

REFERENCES

Bingul, Z., Ertunç, H.M. and Oysu, C. (2005). Applying neural network to inverse kinematic problem for 6r robot manipulator with offset wrist. *7th International Conference on Adaptive and Natural Computing Algorithms*, 112-115.

Daya, B., Khawandi, S. and Akoum, M. (2010). Applying neural network architecture for inverse kinematics problem in robotics. *Journal of Software Engineering and Applications*, *3*, 230-239.

Duka, A.V. (2014). Neural network based inverse kinematics solution for trajectory tracking of a robotic arm. *Procedia Technology, 12,* 20-27.

El-Sherbiny, A., Elhosseini, M.A. and Haikal, A.Y. (2018). A comparative study of soft computing methods to solve inverse kinematics problem. *Ain Shams Engineering Journal, 9(4),* 2535-2548.

Hasan, A.T., Hamouda, A.M.S., İsmail, N. and Al-Assadi, H.M.A.A. (2006). An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 D.O.F serial robot manipulator. *Advances in Engineering Software, 37,* 432-438.

Haykin, S. (1999). Neural networks: A comprehensive foundation, 2nd ed., Prentice Hall, New Jersey.

Jiang, Z.H. and Ishita, T. (2008). A neural network controller for trajectory control of industrial robot manipulators. *Journal of computers, 3(8),* 1-8.

Karlik, B. and Aydin, S. (2000). An improved approach to the solution of inverse kinematics problems for robot manipulators. *Engineering Applications of Artificial Intelligence,13,* 159-164.

Köker, R. (2013). A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Information Sciences, 222,* 528-543.

Köker, R., Öz, C., Çakar, T. and Ekiz, H. (2004). A study of neural network based inverse kinematics solution for a three-joint robot. *Robotics and Autonomous Systems, 49,* 227-234.

Mayorga, R.V. and Sanongboon P. (2005). Inverse kinematics and geometrically bounded singularities prevention of redundant manipulators: An artificial neural network approach. *Robotics and Autonomous Systems, 53,* 164-176.

Tejomurtula, S. and Kak, S. (1999). Inverse kinematics in robotics using neural networks. *Information Sciences, 116,* 147-164.

Xu, Z., Li, S., Zhou, X., Yan, W., Cheng, T. and Huang, D. (2019). Dynamic neural networks based kinematic control for redundant manipulators with model uncertainties. *Neurocomputing, 329,* 255-266.

Zacharie, M. (2012). Advanced Logistic Belief Neural Network Algorithm for Robot Arm Control. *Journal of Computer Science, 8(6),* 965-970.