Journal of Automotive Science and Engineering Technology
University of Ibadan, JASETUI

print-ISSN: 2587-**** https://journals.ui.edu.ng/index.php/JASETUI e-ISSN: 2587-****

Artificial neural network-based inverse kinematics model for trajectory tracking of AR4-MK2 industrial robotic arm

David A. Fadare[1,*], Micheal A. Abubakar[2], Ayodeji Falana[2], Akinola A. Afolabi[1], Dorcas A. Fadare[3]
[1] *Department of Automotive Engineering, Automation, Robotics & Process Simulation Lab., University of Ibadan, Ibadan, Nigeria.*
[2] *Department of Mechanical Engineering, University of Ibadan, Ibadan, Nigeria*
[3] *Faculty of Science, Department of Chemical Sciences, Anchor University, Lagos, Nigeria.*

ABSRACT
Analysis of the inverse kinematics is crucial for robotic arms trajectory tracking control. The main thrust of inverse kinematics analysis is the determination of the robot joint space parameters: rotation angles from the Cartesian space parameters of the end effector position and orientation. Unlike the analysis of direct kinematics, in inverse kinematics, there are no systematic methods for analysing the problem that guarantees closed solutions. The traditional inverse kinematics solution algorithms are often faced with the problem of insufficient generalization, and iterative methods have challenges such as large computation and long solution time. The use of artificial neural networks (ANNs) model can significantly enhance the precision and adaptability of robotic arm control. In this paper, the inverse kinematics analysis of AR4-MK2, a 6-degree-of-freedom articulated desktop-size industrial robot arm is modelled using an artificial neural network (ANN) for trajectory tracking control. The ANN model consists of multi-layered, feed-forward, back-propagation networks with different configurations designed using the Deep Learning Toolbox for MATLAB. The forward kinematics algorithms of the arm developed using the Denavit-Hartenberg method are used to generate the input-output datasets required to train the ANN model. The robot end-effector's Cartesian space parameters including location values (x, y, z) and orientation values $(x_\theta, y_\theta, z_\theta)$) were used as input data, while the joint space parameters consisting of the six (6) joint angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ and θ_6) were used as the output of the network

ARTICLE INFO
Keywords:
Artificial Neural Network
Inverse Kinematics
Trajectory Tracking
AR4-MK2 Robot

1. Introduction

In recent times, the need for accuracy, efficiency, high productivity and cost-effectiveness are major drivers moving the industries towards automation. With the rapid development of emerging technologies such as big data, artificial intelligence, and cloud computing, robotics and artificial intelligence tools have been rapidly developed and widely applied in the context of Industry 4.0 [1–4]. In industrial production, manufacturing processes are gradually moving toward digital production with the rapid adoption of robotics in the manufacturing workspace [5,6]. Beyond the manufacturing workspace, robotics has also continued to find rapid adoption in the healthcare, agriculture, transport, and logistics industries [5, 7]. This innovation has continued to place rigorous and stringent requirements on high-precision control of the robots. Hence, the control of robots has gradually shifted from manual teaching to automated motion control algorithms. As research laboratories continue to explore the world of robotics, there exists a greater necessity for high precision in terms of accuracy, speed, and consistency of motion trajectory tracking and control of the robots [8].

Intelligent robot control relies heavily on the effective implementation of forward kinematics (FK) and inverse kinematics (IK) algorithms [9]. Therefore, the study of motion control algorithms for robots is of great significance to industrial production. Robots are widely used in industrial production, such as mechanical assembly, welding, handling, grasping, and disassembly.

[0]*Corresponding author: David A. Fadare, da.fadare@ui.edu.ng
https://doi.org/**.****/jasetui.2026.0001
Received 20 January 2026; Received in revised form 22 April 2026; Accepted 5 May 2026
Available online 9 June 2026

In the forward kinematics (FK) analysis the end-effector's location in the Cartesian space (or workspace), that is its position and orientation, is determined based on the joint variables, while the inverse kinematics (IK) analysis refers to determination of the of the joint variables that allow the robotic arm to reach the given location and orientation within the workspace. The relationship between forward and inverse kinematics, as well as the relationship between joint angle space and Cartesian coordinate space is shown in Figure 1.

Solving the inverse kinematics problem for articulated robots is a difficult and also quite challenging task [10]. The complexity of this problem is attributed to the robot's geometry and the nonlinear trigonometric equations that describe the mapping between the Cartesian space and the joint space. The complexity of kinematic analysis and control is significantly increased as the number of degrees of freedom (DOFs) of an articulated robotic arm increases. This complexity is primarily attributed to the forward and inverse kinematics analyses required for the robotic arm.

The traditional methods commonly used for the analysis include the algebraic, geometric and iterative methods [11]. However, these traditional methods are known to have inherent setbacks that limit their wide application and acceptability in real industrial settings [12]. The algebraic method is known to have no closed form of solution and is specifically not suited for real (physical) robotic applications, while the geometric method is limited only to analysis involving up to 3 DOF robots. For the Iterative method, the solution is entirely dependent on the starting point, while the Jacobian method does not provide all the solutions when joint velocities of the configurations become large upon following the desired posture in the Cartesian plane [12]. Although a closed-form solution to this problem is preferable in many applications, most of the time this is impossible to determine using the traditional methods. Hence, other alternative methods to determine the solution to the problem have been proposed which include numerical algorithms based on optimization techniques, evolutionary computing or neural networks [13].

Neural networks have long been recognized as being able to represent non-linear relationships that occur between input and output data. Their ability to learn by example makes them a good candidate to provide the mapping between the Cartesian space and the joint space required by the IK problem. ANNs are information processing paradigms inspired by biological nervous systems. ANNs, like human beings, can learn by example, associate data and recall information. As learning in biological systems involves adjustments of synaptic connections that exist between the neurons, so are ANNs made up of simple processing units which are linked by adjustable weight connections to form structures that can learn relationships between sets of variables. After being sufficiently trained, ANNs can perform predictions at very high speeds [14–16]. They can deal with non-linear problems such as multivariate time series prediction. Modelling and predicting experimental data using a neural network-based model gives a more reliable prediction that overcomes the complexity, non–linearity and time-varying characteristics of the pulping operation. Neural network-based models offer exciting advantages, such as learning, adaptation, fault-tolerance, parallelism and generalization.

Several ANN structures used for solving the IK problem include backpropagation-trained feed-forward neural networks or neural networks whose weights are defined in terms of sin and cos to fit the FK representation of the robot [17–20]. Kamlesh Shah et al [21] proposed the application of a Deep Artificial Neural Network (DANN) model to solve the inverse kinematics of a 5-axis serial link robotic manipulator. The joints in this robotic manipulator were rotary. MATLAB was used to train the proposed DANN model.

The goal of this paper is to investigate the use of neural networks for modelling and prediction of the IK solution for a commercially available industrial robot, AR4-MK2, an articulated robot arm with a 6-degree-of-freedom. Maes et al [22] proposed a CAD-based energy-optimal link length geometry and motion profile co-optimization for this industrial robots. However, the application of artificial neural networks to the analysis and solution of complex inverse kinematics problems of the AR4-MK2 industrial robot remains relatively underexplored.

The proposed neural network model was developed with MATLAB and trained using the dataset generated by the FK formulated through the Denavit-Hartenberg (D-H) approach to model the complex inverse-forward mapping of the configuration workspace of the AR4-MK2 industrial robot.

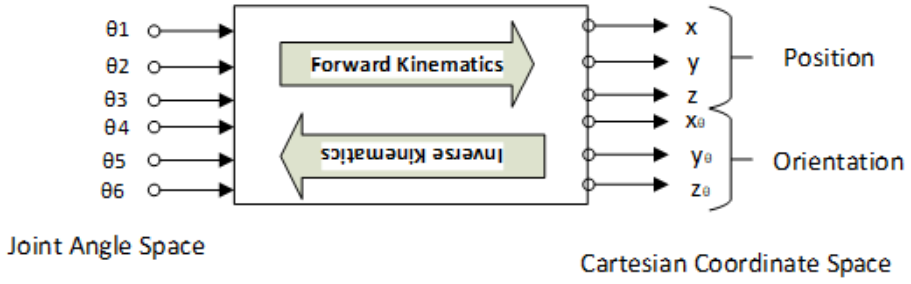


Figure 1: Forward and inverse kinematics relationship.

1.1 Description of the AR4-MK2 robot

The AR4 MK2 is a commercially available 6-degree-of-freedom articulated industrial robot arm developed by Annin Robotics [23]. It's designed to be a low-cost, open-source solution for various applications, including educational purposes, small automation processes, and hobbyist projects. The AR4 MK2 robot is available as a DIY kit, allowing different users to build the robot themselves using aluminium components or 3D-printed parts. It uses an Arduino-based Teensy 4.1 control board, which is compatible with free, open-source Python control software. This robot is widely used in educational programs, small businesses, and by hobbyists for tasks such as inspection, remote TCP, and more. All necessary 3D print files, software, manuals, and source code are available for free on the Annin Robotics website. The joints of the robot are all revolute. The base is associated with the yaw angle, the shoulder, elbow, arm and wrist are associated with the pitch angle and the wrist is also associated with the roll angle. The

computer model of the AR4-MK2 robot is shown in Figure 2.

While the orthographical drawings of the robot are shown in Figure 3. Table 1 shows the basic AR4-MK2 robot parameters. The maximum reach of the robot is 62.9cm in terms of yaw angle and the maximum height (pitch length for the base) is 79.9cm.

2.0 Materials and Methods

2.1 The Forward Kinematics (FK) Model

The joints of the robot are all revolute. The base is associated with the yaw angle, the shoulder, elbow, arm and wrist are associated with the pitch angle and the wrist is also associated with the roll angle. Figure 4 shows the joint configuration of the AR4-MK2 robot. The Denavit-Hartenberg (D-H) method was used to analyze the forward kinematics of the robot. The kinetic structure and the schematics of the robotic arm are shown in Figures 5, respectively. By assigning coordinate frames to the links of the robot, its motion can be analyzed by obtaining a transformation that relates the joint space to the task space. Using transformation matrices $A_i \in \mathbb{R}^{4\times4}$, the relative position and orientation of two link coordinate frames are obtained. The rotation matrix transforms coordinates from the base coordinate system of the robotic arm to the end-effector coordinate system. This transformation method will aid in kinematic analysis. The rotation matrix formula of the computer arm end attitude is as given in Equation 1.

Each Rotation matrix is initialized according to the angle, and the matrix multiplication results are given in Equation 2. The information of this matrix can also be derived from the position and posture of the robotic arm's end-effector. Therefore, through this equation, we can study both the forward and inverse kinematics of the robotic arm. Using the Denavit-Hartenberg (DH) convention, the joint parameters are shown in Table 2.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

$$ {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

Figure 2: The 3D model of the AR4-MK2 robot.

Table 1: The basic AR4-MK2 robot parameters

Parameter	Value	Unit
Mass	1200	kg
Reach	24.75	inches
Payload	4.15	lbs
Repeatability	0.2	mm
Robot weight (aluminium)	27	lbs
Enclosure weight	12.5	lbs
Max. Power Consumption	8.25	amp

2.2 Development of the ANN Model

Deep Learning Toolbox for MATLAB® version 14.2 [24] was used to develop the neural network model. The six (6) basic steps adopted in the development of the ANN model are as follows:

(i) Generation of data;

(ii) Analysis and pre-processing of data;

(iii) Design of the neural network;

(iv) Training and testing of the neural networks;

(v) Simulation and prediction with the neural networks; and

(vi) Analysis and post-processing of predicted results.

2.2.1 Data Generation and Pre-processing

The forward kinematics (FK) model developed using

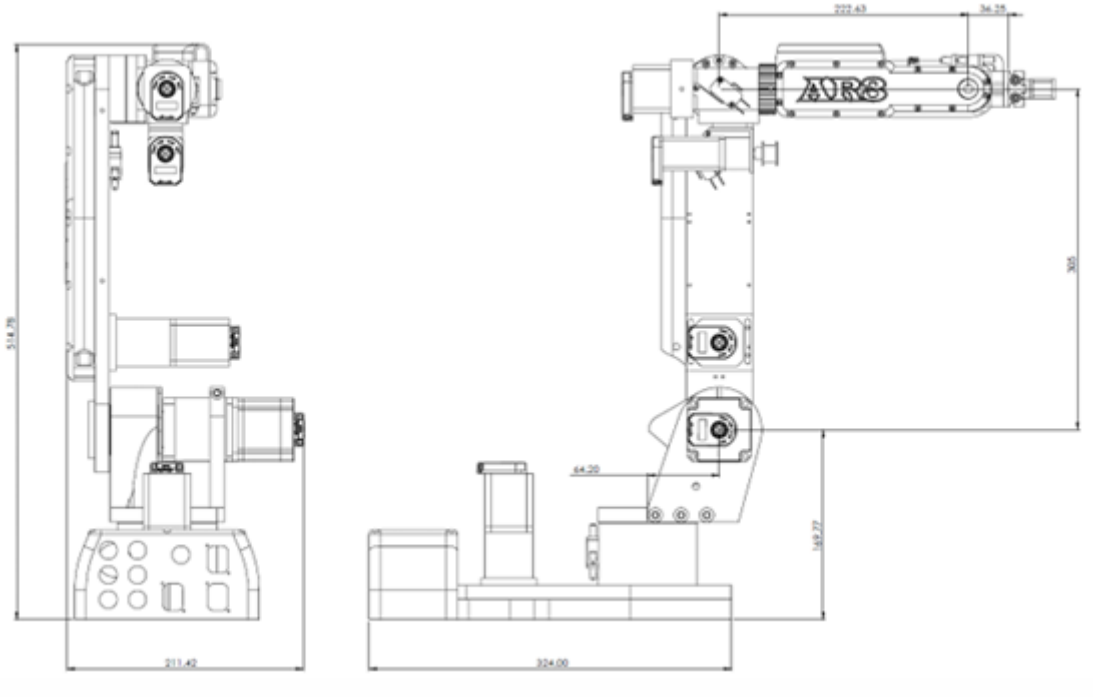


Figure 3: Orthographic drawings of the AR4-MK2 robot.

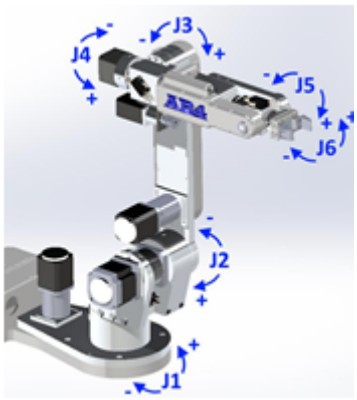
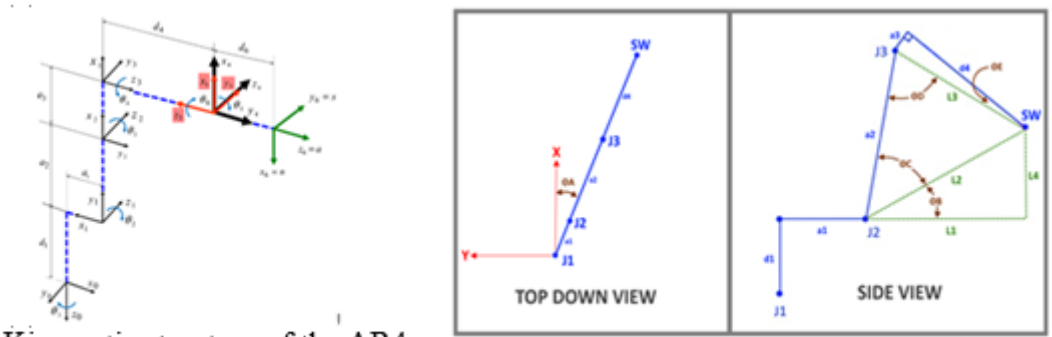


Figure 4: The joint configuration of the AR4-MK2 robot.



(a) Kinematic structure of the AR4-MK2 robot

(b) Schematics of the AR4-MK2 robot

Figure 5: Kinematic structure and schamatics of the AR4-MK2 robot

Table 2: D-H parameters of the AR4-MK2 robotic arm

Joint No.	θ	α (°)	d (mm)	a (mm)
1	J_1	−90	169.8	64.2
2	$J_2 - 90°$	0	0.0	305.0
3	$J_3 + 180°$	90	0.0	0.0
4	J_4	−90	222.6	0.0
5	J_5	90	0.0	0.0
6	J_6	0	36.3	0.0

the Denavit-Hartenberg method is used to generate the input/output datasets required to train the ANN model. The geometry of the robot is defined in terms of its end-effector's joint parameters consisting of the six (6) joint angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ and θ_6) and the Cartesian parameters including the position (x, y, z) and orientation ($x_\theta, y_\theta, z_\theta$)). In the FK model, the joint parameters constitute the input datasets, while the Cartesian parameters constitute the output datasets. Using the FK model, the input/output datasets were generated for different sets of joint angles for any possible configuration of the robot within the robot workspace.

The input/output datasets for 1,000 different configurations of joint angles were generated by varying jointly the joint angles from the lower to upper limits of the angle. Table 3 shows the robot joint parameters including the lower and upper angle limits and the increase in the variations for the six (6) joint angles required to generate the input/output datasets. The input/output datasets generated from the FK model were then used to train the proposed ANN model. The Cartesian space parameters, position sets (x, y, z) and orientation sets ($x_\theta, y_\theta, z_\theta$)) were used as input data, while the joint space parameters, joint angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ and θ_6) were used as the output of the network. Before the training of the network model, the input/output datasets were normalized to values between –1 to +1 using the function *'mapminmax'*. The ith normalized input/output datasets were computed based on Equation 3

$$x_i = 2\left(\frac{d_i - d_{\min}}{d_{\max} - d_{\min}}\right) - 1 \tag{3}$$

Where x_i is the i^{th} normalized input/output data set, d_i is the i^{th} raw input/output dataset, while dmin and dmin are the minimum and maximum raw input/output dataset.

Subsequently, the original input/output datasets were recovered from the proposed ANN model through the use of the function 'mapminmax.reverse'. The input/target dataset was divided randomly into two subsets: training and testing datasets. The training set, which consists of $\frac{2}{3}$ of the dataset, was used for computing the gradient and updating the network weights and biases, while $\frac{1}{3}$ of the dataset was used as a test dataset.

2.2.2 Design of the ANN Model

The Deep Learning Toolbox for MATLAB was used to design a standard multi-layered feed-forward back-propagation network model using the function 'newff". The network models consist of three layers: input layer; hidden layer; and output layer. A typical multi-layered ANN configuration is shown in Figure 8. There were six (6) input parameters in the network, which consisted of the Cartesian space parameters, position sets (x, y, z) and orientation sets ($x_\theta, y_\theta, z_\theta$))) and six (6) output parameters corresponding to the joint space parameters, joint angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ and θ_6).

2.2.3 Design of Network Structure

Determining the optimum network structure can be a very complicated process. Having the optimum number of hidden layers and number of neurons in the layers is essential to ensure the great accuracy of the ANN model and achieve the minimum possible error in the output. Different network structures with single and double hidden layer topologies were investigated with the number of neurons varied from 5-20, at intervals of 5 neurons. In the input layer, neurons with no activation function were used but neurons with the log-sigmoid (logsig) or hyperbolic tangent sigmoid (tansig) activation functions were used in the hidden layer(s). The log-sigmoid function takes neuron input and returns an output squashed into the range [0,1] using the algorithm in Equation 4.

$$f(x) = \frac{1}{1 + \exp(-n)} \tag{4}$$

The hyperbolic tangent sigmoid activation function takes the neuron input and gives a result on the neuron output in the range [-1,1] based on the equation given in Equation 5..

$$f(x) = \frac{2}{1 + \exp(-2n)} - 1 \tag{5}$$

These activation functions were used in the design of the ANN model because they gave the best learning results and are suitable for solving similar problems, are the most commonly used and well-proven in Deep learning. In the output layer, neurons with the linear activation function 'pureln' were used. The linear activation simply returns the input x as the output as defined in Equation 6:

$$f(x) = x \tag{6}$$

2.2.4 Design of Network Structure

The Levenberg-Marquardt (LM) and Scaled conjugate gradient (SCG) learning algorithms were used for the training of the networks. To avoid 'overfitting' of the data and hence improve generalization of the network the 'early stopping' technique was used in conjunction with the training algorithms. The input/target dataset was divided randomly into two subsets: training and testing datasets. The training set, which consists of $\frac{2}{3}$ of the dataset, was used for computing the gradient and updating the network weights and biases, while $\frac{1}{3}$ of the dataset was used as validation and test dataset respectively for each network.

The maximum number of 500 epochs, learning rate of 0.5 and network performance goal of MSE $= 1.0 \times 10^{-10}$ were used in the training process. The different network structures were trained using the 1,000 input/output datasets and their predictive performance was evaluated based on the Mean Square Error (MSE) and Mean Absolute Percentage Error (MAPE) between the predicted and the actual values. During the training process, the 'weights' and 'biases' of the network are adjusted to minimize the mean square error (MSE) between the actual

and the predicted values. The mean square error was computed as in Equation 7.

$$\text{MSE} = \frac{1}{Q}\sum_{k=1}^{Q} e(k)^2 = \frac{1}{Q}\sum_{k=1}^{Q}(t(k) - a(k))^2 \quad (7)$$

Where, Q is the number of the input/output dataset, e(k) is the network error, t(k) is the experimental value and a(k) is the network predicted value.

The training was terminated when the performance goal of MSE $< 1.0 \times 10^{-10}$ or when the maximum number of iterations was equal to 500. The performance of the model was tested based on the coefficient of determination (R-value), MSE, and the mean absolute percentage error (MAPE) between the predicted and the actual values for the entire, training and test datasets. The MAPE was computed as in Equation 8

$$\text{MAPE} = \frac{1}{Q}\sum_{k=1}^{Q}\frac{|t(k) - a(k)|}{t(k)} \times 100\% \quad (8)$$

2.2.5 Selection of optimum network topology

The analysis of the network performance (%) across different network configurations, focusing on the training algorithm, number of hidden layers, number of neurons and activation function across the training and testing datasets was carried out. The optimum network structure was selected based on minimum values of MSE and MAPE between the actual and model prediction using both the training and testing datasets.

Table 3: Robot joint parameters (lower and upper angles and increment)

Joint Number	Lower Angle (degree)	Upper Angle (degree)	Increment (degree)
Joint 1 (Base rotation)	-170	170	0.34
Joint 2 (Shoulder rotation)	-42	90	0.132
Joint 3 (Elbow rotation)	-89	52	0.141
Joint 4 (Arm rotation)	-165	165	0.33
Joint 5 (Wrist pitch)	-105	105	0.21
Joint 6 (Wrist roll)	-155	155	0.31

3.0 Results and Discussion

3.1 Optimization of Network Performance

A total of 32 different network structures with single and double hidden layers with varying numbers of neurons from 5–20, at intervals of 5 neurons with two (2) types of activation functions, trained using two (2) training algorithms were investigated. The network performances in terms of the MAPE, MSE and R-value between the actual and ANN predicted values for the training and testing datasets are shown in Table 4. The progression in the network performance (MSE) during the training process is shown in Figure 6. The correlation coefficient (R-value) between the predicted and the actual values for the training, testing and whole datasets are shown in Figure 7.

Based on the R-values, as it can be seen (Figure 7) that the model has a predictive performance not less than R = 1 (100% correlation) for both training, testing and whole datasets. This indicates the generalization capability of the proposed model for inverse kinematic analysis of robotic arms. Kumar and Chand [25] have reported a 100% correlation for the training dataset for artificial neural network-based inverse kinematic analysis of the SCORBT-ER 4u robotic arm.

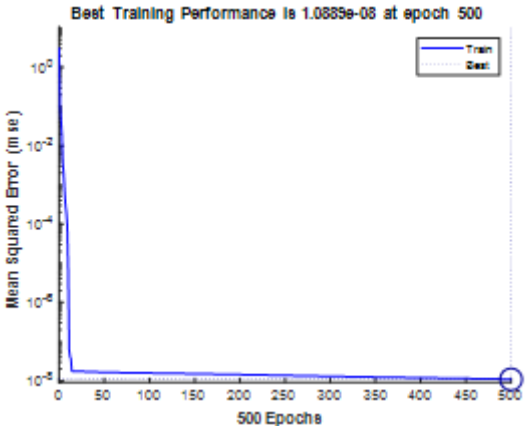
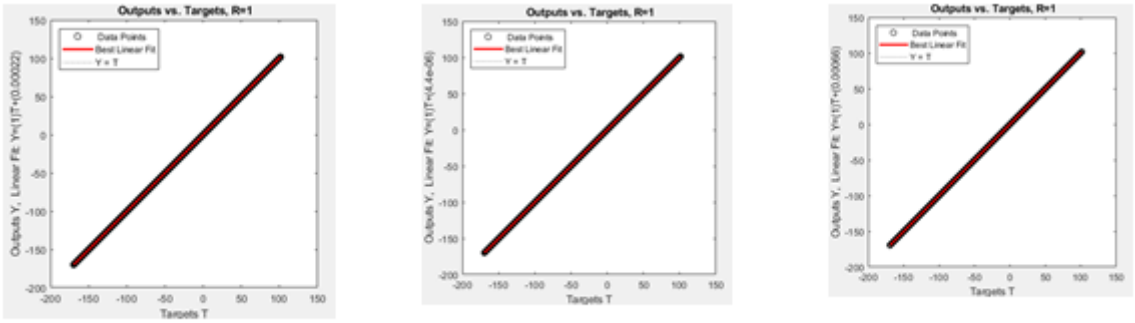


Figure 6: Progression in network performance (MSE) during the training process.

3.1.1 Effect of training algorithmn

The networks trained using the LM algorithm achieved lower MSE values of 1.2917×10^{-4}–1.0000×10^{-3} and 3.0885×10^{-4}–1.4000×10^{-3}, and lower MAPE values of 0.0204–0.1273% and 0.0292–0.1403% for the training and testing datasets, respectively, compared to the SCG algorithm, which yielded MSE values of 3.96×10^{-2}–2.591×10^{-1} and 2.87×10^{-2}–2.795×10^{-1}, and MAPE values of 0.9672–3.3850% and 0.7947–3.3850% for the corresponding training and testing datasets. Based on the MSE and MAPE metrics, the LM-



(a) Whole dataset (b) Training dataset (c) Test dataset

Figure 7: The correlation coefficient (R-value) between the predicted and the actual values.

trained networks exhibited percentage improvements of 99.69% and 97.82%, respectively, on the training dataset, while the corresponding improvements on the testing dataset were 99.59% and 97.44%, respectively.

Based on the MSE and MAPE values for the training dataset, the percentage improvement of networks trained with LM compared with SCG are 99.69 and 97.82%, respectively, while the corresponding values for the testing dataset are 99.59 and 97.44%, respectively. This suggests that LM may be better suited for networks requiring high accuracy. LM's lower error rates indicate that it handles complex relationships in the data more effectively. However, LM is computationally intensive, so it might be best suited for cases where computation time is not a limitation. The SCG training algorithm shows slightly higher error metrics and shorter computation time. It remains a viable choice when computational efficiency is a priority. It's particularly suitable for larger networks where faster convergence is desired, albeit with a slight trade-off in accuracy.

3.1.2 Effect of the number of hidden layers

Networks with double hidden-layer architectures generally exhibited superior performance compared with their single hidden-layer counterparts. Based on the MSE and MAPE values obtained for the training dataset, the percentage improvements achieved by the double hidden-layer networks over the single hidden-layer networks were 12.20% and 2.97%, respectively. Similarly, for the testing dataset, the corresponding improvements were 17.62% and 21.29%, respectively. The enhanced performance can be attributed to the additional hidden layer, which enables the network to learn more complex nonlinear relationships and hierarchical feature representations within the data, thereby reducing prediction errors.

This improvement was particularly evident when the network was trained using the LM algorithm. Although networks trained with the SCG algorithm also benefited from the additional hidden layer, the resulting performance gains were comparatively smaller. This suggests that while increased network complexity can improve predictive accuracy, the LM algorithm is more effective than SCG in exploiting the representational capacity provided by the deeper architecture.

3.1.3 Effect of activation function

The hyperbolic tangent sigmoid (tansig) activation function consistently yields better performance, with lower MSE and MAPE values than log-sigmoid (logsig). Based on the MSE and MAPE values for the training dataset, the percentage improvement of networks with tansig neurons compared with logsig neurons is 24.63 and 10.72%, respectively, while the corresponding values for the testing dataset are 28.58 and 12.93%, respectively. The tansig's range from -1 to 1 gives a smoother gradient that helps networks converge faster, especially when paired with LM, leading to better generalization on the test dataset. This suggests that tansig might be optimal for tasks that require high precision. While logsig has its applications, particularly in binary outputs, it may lead to higher error rates in regression tasks or networks with deeper layers where values outside the 0–1 range are beneficial [26].

3.1.4 Effect of the number of neurons

For both training algorithms, an increase in the number of neurons generally improves performance, reducing MSE and MAPE, up to around 15–20 neurons. An increase in the number of neurons from 5–10 results in network performance improvement of 12.93 and 36.88% in MSE and MAPE, respectively using the training dataset, while the corresponding values for the testing dataset are 45.13 and 46.73%. A further increase in the number of neurons from 10–15 results in a corresponding improvement of 26.30 and 30.57% using the training dataset, while the corresponding values for the testing dataset are 36.04 and 33.25%. For LM networks, configurations with 10–20 neurons strike a balance between complexity and performance, with higher neuron counts often leading to slightly lower errors. In SCG networks, while more neurons initially improve performance, using more than 10 neurons sometimes increases testing MAPE, indicating a risk of overfitting.

3.1.5 Selection of Optimal Network Configuration

As shown in Table 4, Network No. 11, comprising two hidden layers with 15 neurons each and employing the `tansig` activation function (6–15–15–6), trained using the LM algorithm, achieved the lowest training errors, with an MSE of 1.2917×10^{-4} and a MAPE of 0.0204%. The network also recorded the lowest testing errors, with an MSE of 3.0885×10^{-4} and a MAPE of 0.0292%. Consequently, this network architecture was selected as the optimum model for the study (Figure 8). The superior performance of this configuration demonstrates its effectiveness in balancing model complexity and prediction accuracy among all the network structures investigated.

Table 4: Performance comparison of ANN models

Network No.	Training Algorithm	Structure of Hidden Layer			Training Dataset			Testing Dataset		
		No. of Layer	Activation Function	No. of Neurons	MSE	MAPE (%)	R-value	MSE	MAPE (%)	R-value
1	LM	1	tansig	5	5.2098E-04	0.0549	1.0	6.0411E-04	0.0548	1.0
2	"	"	"	10	2.9950E-04	0.0448	1.0	4.7003E-04	0.0444	1.0
3	"	"	"	15	2.8534E-04	0.0335	1.0	4.4941E-04	0.0495	1.0
4	"	"	"	20	3.2986E-04	0.0395	1.0	4.8271E-04	0.0400	1.0
5	"	1	logsig	5	1.0000E-03	0.1273	1.0	1.4000E-03	0.1403	1.0
6	"	"	"	10	3.7402E-04	0.0383	1.0	5.0565E-04	0.0521	1.0
7	"	"	"	15	3.9498E-04	0.0416	1.0	5.1924E-04	0.0520	1.0
8	"	"	"	20	3.0553E-04	0.0342	1.0	4.6827E-04	0.0425	1.0
9	"	2	tansig	5	5.6462E-04	0.0689	1.0	6.5595E-04	0.0666	1.0
10	"	"	"	10	2.1221E-04	0.0337	1.0	4.1449E-04	0.0349	1.0
11	"	"	"	15	1.2917E-04**	0.0204**	1.0	3.0885E-04*	0.0292*	1.0
12	"	"	"	20	2.2825E-04	0.0306	1.0	4.3029E-04	0.0342	1.0
13	"	2	logsig	5	5.6486E-04	0.0830	1.0	6.3805E-04	0.0715	1.0
14	"	"	"	10	3.2269E-04	0.0322	1.0	4.5157E-04	0.0387	1.0
15	"	"	"	15	2.5698E-04	0.0264	1.0	4.3496E-04	0.0431	1.0
16	"	"	"	20	2.4824E-04	0.0280	1.0	4.3011E-04	0.0298	1.0
17	SCG	1	tansig	5	1.1420E-01	2.2401	1.0	1.5840E-01	2.2401	1.0
18	"	"	"	10	9.6000E-02	1.8768	1.0	1.2900E-01	1.8768	1.0
19	"	"	"	15	6.7800E-02	1.5524	1.0	9.9900E-02	1.5524	1.0
20	"	"	"	20	1.6660E-01	2.7526	1.0	1.7970E-01	2.7526	1.0
21	"	"	logsig	5	2.1910E-01	3.3850	1.0	2.2110E-01	3.3850	1.0
22	"	"	"	10	9.7900E-02	1.2031	1.0	9.3800E-02	1.2031	1.0
23	"	"	"	15	1.1170E-01	1.8121	1.0	1.1670E-01	1.8121	1.0
24	"	"	"	20	6.4300E-02	1.8157	1.0	6.6000E-02	1.8157	1.0
25	"	2	tansig	5	1.5060E-01	1.8228	1.0	1.5850E-01	2.0903	1.0
26	"	"	"	10	1.3950E-01	1.7800	1.0	1.0970E-01	1.3191	1.0
27	"	"	"	15	2.5910E-01	3.2673	1.0	2.7950E-01	3.1342	1.0
28	"	"	"	20	1.5140E-01	2.6758	1.0	1.4620E-01	2.3584	1.0
29	"	"	logsig	5	8.0600E-02	1.6864	1.0	1.2090E-01	2.0219	1.0
30	"	"	"	10	3.9600E-02	0.9672	1.0	2.8700E-02	0.7947	1.0
31	"	"	"	15	6.8600E-02	1.8537	1.0	6.9500E-02	1.3642	1.0
32	"	"	"	20	1.8000E-01	3.1980	1.0	1.8170E-01	2.5245	1.0

LM: Levenberg-Marquardt
SCG: Scaled conjugate gradient
** Minimum values of MSE and MAPE based on the training dataset
* Minimum values of MSE and MAPE based on the testing dataset

Table 5: The joint angle parameters and the MSE of each joint.

Joint Number	Lower Angle (°)	Upper Angle (°)	Increment (°)	MSE
Joint 1 (Base rotation)	-170	170	0.34	7.2242×10^{-4}
Joint 2 (Shoulder rotation)	-42	90	0.132	1.1592×10^{-4}
Joint 3 (Elbow rotation)	-89	52	0.141	1.2524×10^{-4}
Joint 4 (Arm rotation)	-165	165	0.33	7.0856×10^{-4}
Joint 5 (Wrist pitch)	-105	105	0.21	2.9482×10^{-4}
Joint 6 (Wrist roll)	-155	155	0.31	6.1479×10^{-4}

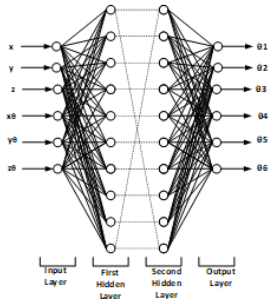


Figure 8: A typical multi-layered ANN configuration with 6–10–10–6.

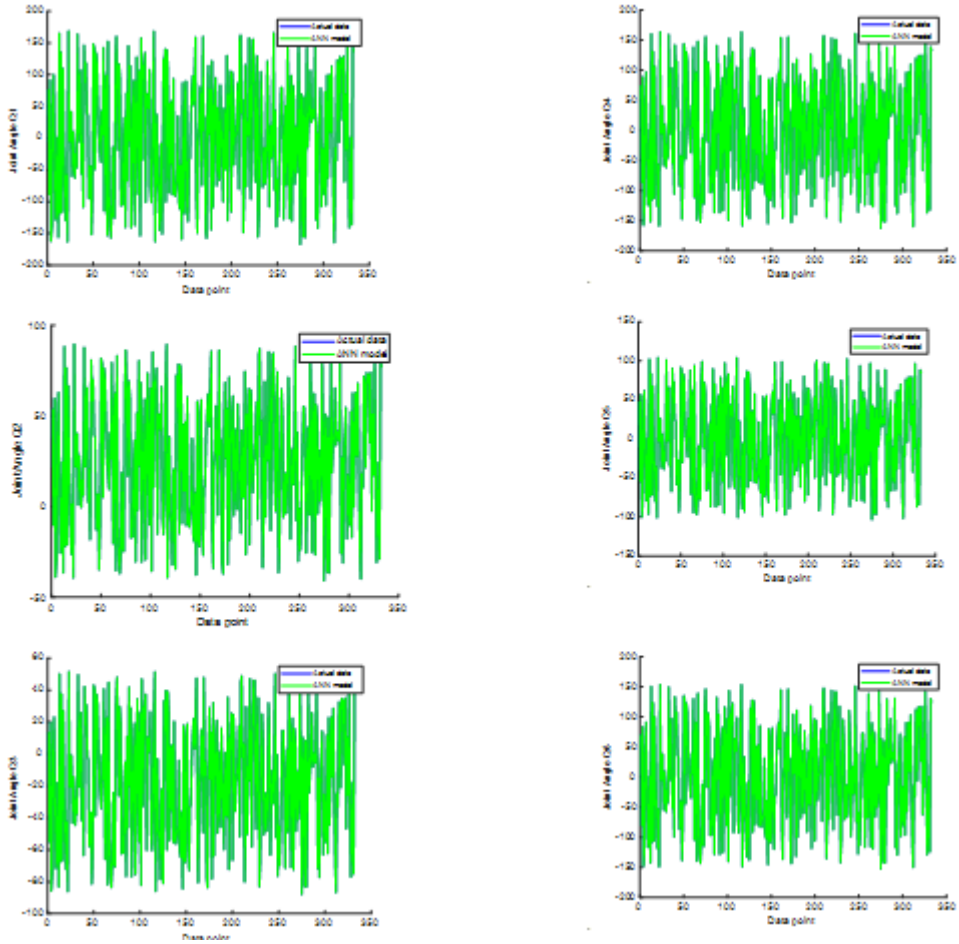


Figure 9: Graphical result of the actual and ANN prediction for joint angles.

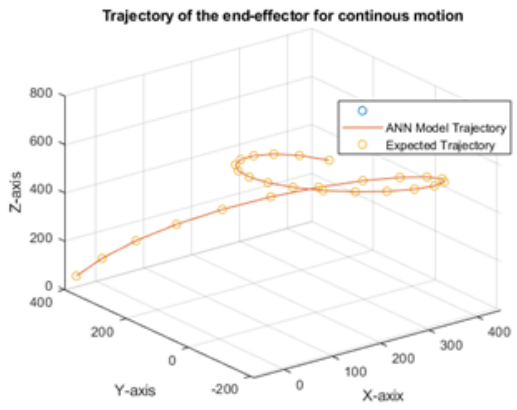


Figure 10: Trajectory tracking of the end effector for continuous motion.

3.2 Prediction of Robot Joint Angles

After selecting the optimal network architecture and training it to an acceptable level of accuracy, the model was employed to predict the six joint angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$, and θ_6). The inputs to the model were the end-effector's Cartesian position (x, y, z) and orientation $(x_\theta, y_\theta, z_\theta)$. The predictive performance of the ANN for the six joint angles using the testing dataset is presented in Figure 9. In the figure, the blue line represents the actual values, while the green line represents the ANN-predicted values. As observed, the predicted and actual responses closely overlap, indicating a high level of agreement between the model outputs and the target data.

Table 5 shows the joint angle parameters and the MSE between the model prediction and the actual values of the six (6) joint angles using the testing dataset. From Table 5 it can be seen that there exists a direct correlation between the MSEs and the increments in each joint angle used in generating the input/output datasets in this study. Joints with higher increments have higher MSE. Joints 2 and 3 have lower MSE and increments, suggesting better control and precision in these joints. Hence, for better control and precision of the joints small increments or fine intervals are required in the joint angle during the process of dataset generation.

To verify the accuracy and precision of the proposed ANN model for trajectory tracking and control, the model was tested to perform continuous motion based on the given expected trajectory. The Cartesian space parameters of the expected trajectory were used as input to the proposed ANN model to obtain the joint angles. The model-predicted joint angles were then transformed by the D-H forward kinematics algorithm to obtain the end effector position and orientation. The ANN model predicted trajectory and the expected trajectory are shown in Figure 10. As seen in the figure, the trajectory obtained based on the ANN model is almost identical to the expected trajectory.

4.0 Conclusion

In this study, it was demonstrated that the multi-layered ANN model is viable for solving the inverse kinematics problem of the AR4-MK2 robot, a commercially available 6-degree-of-freedom articulated industrial robot. The ANN model solves the problem of finding the robot joint angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ using the end-effector Cartesian space parameters: position (x, y, z) and orientation $(x_\theta, y_\theta, z_\theta)$ as the input dataset.

The proposed model achieved a predictive performance of $R = 1$ (100% correlation) for the training, testing, and entire datasets.

The selected optimal network structure consists of two hidden layers with 15 neurons each, using the *tansig* activation function (6–15–15–6), trained using the Levenberg–Marquardt (LM) algorithm. This configuration produced the lowest mean squared error (MSE) of 1.2917×10^{-4} and mean absolute percentage error (MAPE) of 0.0204% for the training dataset, and corresponding MSE of 3.0885×10^{-4} and MAPE of 0.0292% for the testing dataset.

The proposed ANN model is referred to as *AR4-MK2NeuralNetwork 1.0v*.

References

[1] M. Javaid, A. Haleem, R. P. Singh, and R. Suman, "Artificial intelligence applications for industry 4.0: A literature-based study," *Journal of Industrial Integration and Management*, vol. 7, no. 01, pp. 83–111, 2022.

[2] D. Mathew, N. C. Brintha, and J. T. W. Jappes, "Artificial intelligence powered automation for industry 4.0." Springer, 2023, pp. 1–28.

[3] F. A. Alenizi, S. Abbasi, A. H. Mohammed, and A. M. Rahmani, "The artificial intelligence technologies in industry 4.0: A taxonomy, approaches, and future directions," *Computers Industrial Engineering*, vol. 185, p. 109662, 2023.

[4] A. Azizi, *Applications of artificial intelligence techniques in industry 4.0*. Springer, 2019.

[5] D. R. I. Dassanayake, M. K. Buddhika, I. D. K. Maduranga, J. A. Seneviratne, and W. G. C. Kumarage, "Revolutionizing manufacturing: the role of robotics in the 21st century," *Journal of Desk Research Review and Analysis*, vol. 2, no. 1, 2024.

[6] C. Palanisamy, L. Perumal, and C. W. Chin, "A comprehensive review of collaborative robotics in manufacturing," *Engineering, Technology Applied Science Research*, vol. 15, no. 2, pp. 21 970–21 975, 2025.

[7] C. Urrea and J. Kern, "Recent advances and challenges in industrial robotics: A systematic review of technological trends and emerging applications," *Processes*, vol. 13, no. 3, p. 832, 2025.

[8] X. Yin and L. Pan, "Enhancing trajectory tracking accuracy for industrial robot with robust adaptive control," *Robotics and Computer-Integrated Manufacturing*, vol. 51, pp. 97–102, 2018.

[9] N. Manjegowda, Muralidhara, N. R. Jain, and S. K. Shivaswamy, "A comprehensive review of inverse kinematics techniques from analytical foundations to artificial intelligence integration," *International Journal of Advanced Mechatronic Systems*, vol. 12, no. 4, pp. 258–282, 2025.

[10] K. T. Ge, "Solving inverse kinematics constraint problems for highly articulated models," 2000.

[11] S. Kucuk and Z. Bingul, *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK, 2006, vol. 1.

[12] A. Aristidou and J. Lasenby, "Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver," 2009.

[13] A. El-Sherbiny, M. A. Elhosseini, and A. Y. Haikal, "A comparative study of soft computing methods to solve inverse kinematics problem," *Ain Shams Engineering Journal*, vol. 9, no. 4, pp. 2535–2548, 2018.

[14] D. A. Fadare, E. O. Ezugwu, and J. Bonney, "Modeling of tool wear parameters in high-pressure coolant assisted turning of titanium alloy ti6al4v using artificial neural networks," *Pacific Journal of Science and Technology*, vol. 10, no. 2, pp. 68–76, 2009.

[15] D. A. Fadare and O. J. Babayemi, "Modelling the association between in vitro gas production and chemical composition of some lesser known tropical browse forages using artificial neural network," *African Journal of Biotechnology*, vol. 6, no. 18, 2007.

[16] A. Mellit, M. Benghanem, and S. A. Kalogirou, "An adaptive wavelet network model for forecasting daily total solar radiation," *Applied Energy*, vol. 83, pp. 705–722, 2006.

[17] S. K. Nanda, S. Panda, P. R. S. Subudhi, and R. K. Das, "A novel application of artificial neural network for the solution of inverse kinematics controls of robotic manipulators," *International Journal of Intelligent Systems and Applications*, vol. 4, no. 9, pp. 81–91, 2012.

[18] R. Gao, "Inverse kinematics solution of robotics based on neural network algorithms," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 12, pp. 6199–6209, 2020.

[19] A. R. J. Almusawi, L. C. Dülger, and S. Kapucu, "A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242)," *Computational intelligence and neuroscience*, vol. 2016, no. 1, p. 5720163, 2016.

[20] D. Cagigas-Muñiz, "Artificial neural networks for inverse kinematics problem in articulated robots," *Engineering Applications of Artificial Intelligence*, vol. 126, p. 107175, 2023.

[21] S. Kamlesh Shah, R. Mishra, and L. Samprit Ray, "Solution and validation of inverse kinematics using deep artificial neural network," *Materials Today: Proceedings*, vol. 26, pp. 1250–1254, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785320310038

[22] B. Maes, N. V. Oosterwyck, R. D. Laet, A. Chevalier, and S. Derammelaere, "Cad-based energy-optimal link length geometry and motion profile co-optimization for multi-actuated serial robots," in *2025 13th International Conference on Control, Mechatronics and Automation (ICCMA)*, 2025, pp. 142–148.

[23] C. Annin, "Annin robotics the ar4 a low-cost, open-source 6dof industrial robot," Tech. Rep., 2025 (www.anninrobotics.com).

[24] The MathWorks, Inc., *Neural Network Toolbox Version 5 for MATLAB 6.1 (R12.1)*, The MathWorks, Inc., 2001.

[25] R. R. Kumar and P. Chand, "Inverse kinematics solution for trajectory tracking using artificial neural networks for scorbot er-4u." IEEE, 2015, pp. 364–369.

[26] A. Wanto, A. P. Windarto, D. Hartama, and I. Parlina, "Use of binary sigmoid function and linear identity in artificial neural networks for forecasting population density," *IJISTECH (International Journal of Information System and Technology)*, vol. 1, no. 1, pp. 43–54, 2017.