IJRCS
International Journal of Robotics and Control Systems
Vol. 3, No. 2, 2023, pp. 330-353
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Forward and Inverse Kinematics Solution of A 3-DOF Articulated Robotic Manipulator Using Artificial Neural Network

Abdel-Nasser Sharkawy [a, b,1, *], Shawkat Sabah Khairullah [c,2]

[a] Mechanical Engineering Department, Faculty of Engineering, South Valley University, Qena 83523, Egypt
[b] Mechanical Engineering Department, College of Engineering, Fahad Bin Sultan University, Tabuk 47721, Saudi Arabia
[c] Department of Computer Engineering, College of Engineering, University of Mosul, Mosul, Iraq
[1] abdelnassersharkawy@eng.svu.edu.eg; [2] shawkat.sabah@uomosul.edu.iq
* Corresponding Author

ARTICLE INFO

Article history
Received March 26, 2023
Revised May 01, 2023
Accepted May 09, 2023

Keywords
Artificial Neural Network;
biological;
Forward Kinematics;
Inverse Kinematics;
3-DOF Robot;
Sinusoidal Motion

ABSTRACT

In this research paper, the multilayer feedforward neural network (MLFFNN) is architected and described for solving the forward and inverse kinematics of the 3-DOF articulated robot. When designing the MLFFNN network for forward kinematics, the joints' variables are used as inputs to the network, and the positions and orientations of the robot end-effector are used as outputs. In the case of inverse kinematics, the MLFFNN network is designed using only the positions of the robot end-effector as the inputs, whereas the joints' variables are the outputs. For both cases, the training of the proposed multilayer network is accomplished by Levenberg Marquardt (LM) method. A sinusoidal type of motion using variable frequencies is commanded to the three joints of the articulated manipulator, and then the data is collected for the training, testing, and validation processes. The experimental simulation results demonstrate that the proposed artificial neural network that is inspired by biological processes is trained very effectively, as indicated by the calculated mean squared error (MSE), which is approximately equal to zero. The resulted in smallest MSE in the case of the forward kinematics is 4.592×10^{-8} in the case of the inverse kinematics, is 9.071×10^{-7}. This proves that the proposed MLFFNN artificial network is highly reliable and robust in minimizing error. The proposed method is applied to a 3-DOF manipulator and could be used in more complex types of robots like 6-DOF or 7-DOF robots.

1. Introduction

In the research literature review [1]–[3], the positions and orientations of the end-effector in the forward kinematics problem are determined using the joints' variables. This process is straightforward, easy, and not complex. This part is very important and crucial for calculating the error of the position and/or the orientation to calculate the controller's qualify [4]. In contrast, inverse kinematics is opposite to forward kinematics in such a way that the joints' variables can be calculated using known positions and orientations of the robot end-effector. This process is complex, difficult, and computationally expensive [3]. This part is very important for determining the robot's motion to reach the desired position and for programming the manipulator for performing tasks.

For the forward kinematics problem, different types of solutions were considered by researchers: 1) Denavit and Hartenberg (DH), 2) product of exponential (POE), 3) dual-quaternion, 4) geometric approach, and 5) machine learning-based approaches as support vector regression (SVR) and neural network (NN), as shown in Fig. 1. Denavit and Hartenberg in [5] proposed a new method including four parameters that are considered necessary for the transformation procedure that occurs between two joints. All the parameters have been named as the D and H parameters. Furthermore, the four parameters were considered the operational standard for describing the kinematics of the robot. Although the presented D and H method is the most concise method compared with all, it contains some limitations, as described in [6]. For example, Asif and Webb in [7] performed the forward kinematics of a 6-DOF articulated robot with a spherical wrist by the use of the DH parameters. The product of exponential and unit dual quaternion was proposed for forward kinematics problems [8], [9]. In [10], a comparison research work was developed between the POE method and the unit dual quaternion in determining the forward motion of the 7-DOF KUKA LWR robot. Their experimental results proved the unit dual quaternion has a higher compactness compared with the product of the exponential method. In addition, the unit dual quaternion facilitates better comprehension of the geometrical purpose of the joint axes. The geometric approach has also been proposed for the forward motion problem. In [11], Kim et al. used the conformal geometric algebra for utilizing three SPS/S to analyze the forward kinematics of the duplicated motion manipulator. An additional sensor was required to provide more positional information and to allow the unique solution to be chosen geometrically from the many found solutions using the geometric approach. SVR [12] is a supervised machine learning method used widely in regression tasks. NN [13]–[15] has the ability to approximate any linear/nonlinear function and generalize it in the case of different conditions. SVR was used with the forward motion of parallel manipulator robots [16]–[19].

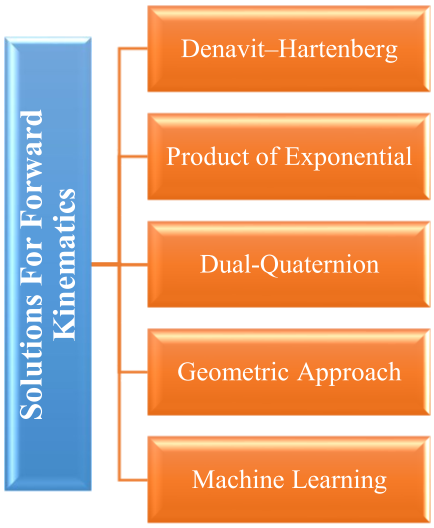


Fig. 1. The different types of solutions used for the forward kinematics problem of a robot

For the inverse kinematics problem, also different types of solutions were considered by researchers, such as 1) closed-form solution method, 2) numerical solution method, 3) evolutionary computing, and 4) neural networks (NNs), as shown in Fig. 2. Closed-form solution method was used when the analytic expression or the polynomial has less than 4-DOF [2]. This solution depended on robot-specific geometry for the formulation of the mathematical model. An example of this approach was presented in [20], where Lou et al. used the closed-form solution for the inverse kinematics of the manipulator based on the general spherical joint. Numerical methods were proposed for the kinematics problem in the inverse mode in case of the resulting polynomial in the solution has more than 4-DOF [21]. These mentioned methods are less accurate compared to the methods of closed-form [2]. An example of these methods was presented in [22], where Elnady developed an iterative technique to solve the inverse kinematics. Evolutionary computing was proposed in ref. [23], [24] as a method for inverse kinematics solution. NNs were also developed to solve the problem of inverse kinematics. Tejomurtula and Kak [25] presented the structured NNs for inverse kinematics. The conventional backpropagation algorithm was performed for the NN training. This led to some difficulties regarding accuracy. NNs were also used in ref. [3], [26], [27].

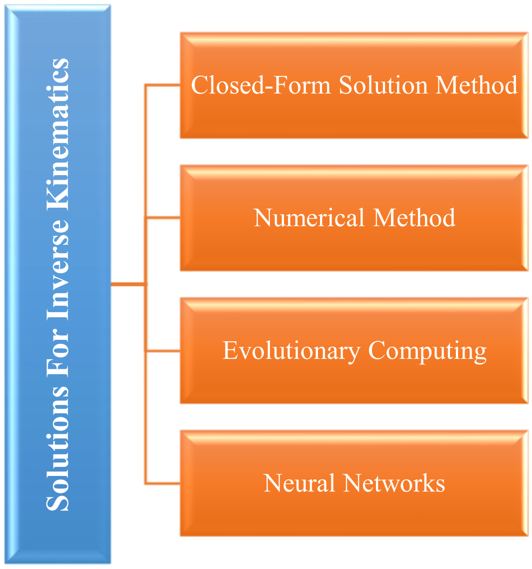


Fig. 2. The different types of solutions used for the inverse kinematics problem of a robot

From the above discussion, further investigation is recommendable for using the NN as a simple and intelligent technique compared with other methods for forward and particularly inverse kinematics solutions. Achieving high levels of performance and reliability of the NN is required by having small values of mean squared error (MSE), which are close to zero and the training error. This can increase the accuracy and robustness of the application and estimate the forward and inverse kinematics correctly. In addition, minimizing the input size of the implemented NN is also recommendable to minimize the complexity and mathematical computations.

The primary contribution of our research paper is proposing an MLFFNN network to solve the issue of kinematics for the 3-DOF articulated robotic manipulator in both forward and inverse modes. In forward kinematics used in the design of the network, the positions and orientations for the manipulator end-effector are outputs, and the joints' variables for the manipulator are used as the inputs. For the inverse kinematics case, only the positions of the end-effector are used as the inputs of the implemented MLFFNN to minimize the size of inputs. The joints' variables are the outputs. In each case, the proposed network is trained using LM learning, which provides fast convergence easily. The main concern during the training on the MLFFNN is obtaining a very small (close to zero) MSE and training error. The training is executed in MATLAB using collected data considering the sinusoidal joints' motion of the manipulator. The testing and the verification of the trained MLFFNN are presented to investigate its reliability in minimizing the approximation error and its effectiveness in estimating the forward and inverse kinematics correctly.

The rest of this paper is divided into the following sections. Section 2 illustrates the forward kinematics of the 3-DOF articulated robot using DH parameters and the inverse kinematics depending on the geometrical approach. In Section 3, the collected data considering sinusoidal joints' motion are described. The design, analysis, and testing of the proposed network to resolve the kinematics problem in forward mode are explained in Section 4. In Section 5, the architectural concept, training, testing and verification of the proposed MLFFNN for solving the problem of inverse kinematics are presented. Finally, Section 6 summarizes the work shown in this manuscript, and Section 7 gives some points for work in the future.

2. Kinematics of a 3-DOF Articulated Manipulator in the Forward and Reverse Mode

This section discusses the forward kinematics as well as the inverse kinematics of the 3-DOF articulated (with rotational joints) manipulator. This type of robot is presented in Fig. 3. In Subsection 2.1, parameters of Denavit-Hartenberg (DH) are used to illustrate the concept of forward kinematics, whereas a geometrical method is used to show the concept of inverse kinematics in Subsection 2.2.

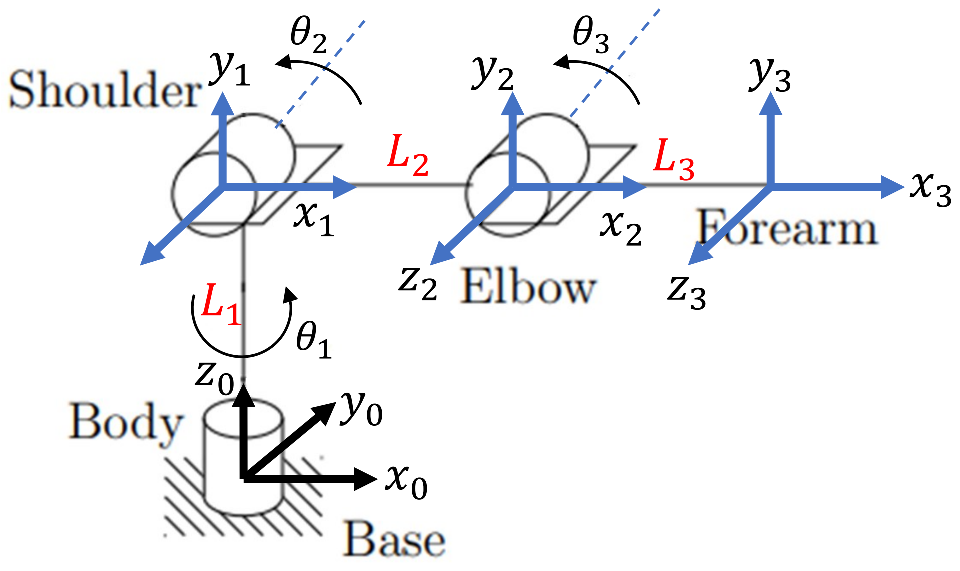


Fig. 3. The 3-DOF articulated robotic manipulator [28], [29]

2.1. Forward Kinematics

The forward kinematics of the 3-DOF manipulator is performed using the DH parameters, which are considered the standard for robot kinematics description. These parameters are presented in Table 1. In the table, a is the link length, α is the link twist, d is the link offset, and θ is the joint angle.

Table 1. The DH parameters of the 3-DOF articulated manipulator

Link	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	90°	L_1	θ_1
2	L_2	0	0	θ_2
3	L_3	0	0	θ_3

According to Table 1, the position and orientation of the robot end-effector are represented by a whole homogeneous transformation matrix, which is obtained as Equation (1) [8].

$$T_0^3 = \begin{bmatrix} C_1C_{23} & -C_1S_{23} & S_1 & L_3C_1C_{23} + L_2C_1C_2 \\ S_1C_{23} & -S_1S_{23} & -C_1 & L_3S_1C_{23} + L_2S_1C_2 \\ S_{23} & C_{23} & 0 & L_3S_{23} + L_2S_2 + L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & p_x \\ R_{21} & R_{22} & R_{23} & p_y \\ R_{31} & R_{32} & R_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

where, p_x, p_y, and p_z represent the position of the robot end-effector in $x-$, $y-$, and $z-$ directions. R_{11}, R_{12}, R_{13},……. R_{33} illustrate the orientation of the robot end-effector. $C_1 = \cos(\theta_1)$, $S_1 = \sin(\theta_1)$, $C_{23} = \cos(\theta_2 + \theta_3)$, and $S_{23} = \sin(\theta_2 + \theta_3)$.

2.2. Inverse Kinematics

In this subsection, the inverse kinematics (the joints' variables θ_1, θ_2, and θ_3) are determined using the geometrical approach. This approach solves the joint variable θ_i by projecting the robot onto the plane of $x_{i-1} - y_{i-1}$ and then solving a simple problem of trigonometry.

The joints' variables using this approach are given as Equations (2), (3), and (4) [29].

$$\theta_1 = \arctan\left(p_y, p_x\right) \tag{2}$$

where, θ_1 is defined only when $p_x^2 + p_y^2 > 0$.

$$\theta_3 = atan2(\sin\theta_3\,, \cos\theta_3) \tag{3}$$

where, $\cos\theta_3 = \frac{p_x^2+p_y^2+(p_z-1)^2-L_2^2-L_3^2}{2L_2L_3}$ and $\sin\theta_3 = \pm\sqrt{1-(\cos\theta_3)^2}$. $\sin\theta_3 \in [-0.173, 0.173]$ (Otherwise, the end-effector's point is outside the work area). This happens because of the constraint of joint three.

$$\theta_2 = atan2(\sin\theta_2\,, \cos\theta_2) \tag{4}$$

where, θ_2 is defined only when $p_x^2 + p_y^2 + (p_z - 1)^2 > 0$.

$\sin\theta_2 = \frac{(L_2+L_3\cos\theta_3)(p_z-L_1)-L_3\sin\theta_3(p_x\cos\theta_1+p_y\sin\theta_1)}{L_3^2(\sin\theta_3)^2-(L_2+L_3\cos\theta_3)^2}$ and $\cos\theta_2 = \pm\sqrt{1-(\sin\theta_2)^2} \in [-1, 1]$.

It is clear from equations (2)-(4) the solution of inverse kinematics, depending on the geometrical approach, is complex and more cumbersome. Furthermore, this is familiar in robotics books that the inverse kinematics solution is complex. Therefore, an MLFFNN network is used for solving the kinematics of the 3-DOF articulated manipulator in forward and inverse. MLFFNN has the following properties and advantages:

1) It is a very simple architecture in comparison to the other different NNs types [13], [30], [31].
2) It has the ability of adaptivity, parallelism, and generalization [32]–[34]. In addition, it can be linear or nonlinear.
3) It is applied successfully in various types of engineering problems [35]–[40].

However, the MLFFNN network needs a high number of input and target pairs in the training stage [41], [42]. This disadvantage is taken into consideration in the current research, and we use large datasets. However, overfitting is avoided, as seen from the results.

In this current research work, the developed MLFFNN is trained depending on the Levenberg-Marquardt (LM) learning algorithm, which possesses the following qualities:

1) The work can be implemented quickly with the mentioned learning algorithm. It is an optimization technique with a second order that approximates Newton's Method and has a solid theoretical foundation as well as quick convergence [43], [44].
2) This algorithm is preferred because it strikes a balance between the gradient descent algorithm's assured convergence and the quick learning speed of the traditional Newton's method [43], [45]. In addition, its preference for large datasets and its convergence in fewer iterations and a short time in comparison with the other learning methods [31].

Using the LM algorithm, the adjustment of the weight $\Delta\boldsymbol{w}$, which is applied to the parameter vector $\boldsymbol{w}$, is calculated using Equation (5) [13], [31],

$$\Delta\boldsymbol{w} = [\boldsymbol{H} + \boldsymbol{\beta I}]^{-1}\boldsymbol{G} \tag{5}$$

where $\mathbf{H}$ and $\boldsymbol{G}$ represent, respectively, the Hessian and the gradient vector of the second-order function. $\mathbf{I}$ is the identity matrix which, its dimension is the same dimensions as $\mathbf{H}$. $\boldsymbol{\beta}$ is a regularizing parameter responsible for forcing the part $(\mathbf{H}+\boldsymbol{\beta}\mathbf{I})$ to be positive definite as well as safely well-conditioned through the computation.

The next section discusses in detail the generated data that have been used for the analysis, testing, and validation of the MLFFNN network.

3. Generated Data

This section describes the data that have been utilized for validating and testing the proposed implemented MLFFNN network to solve the forward and inverse kinematics. A sinusoidal motion is chosen to be commanded to the three joints of the articulated robotic manipulator. This motion is given by Equation (6).

$$\theta_i(t) = \frac{\pi}{4} - \frac{\pi}{4} cos\,(2\pi f t) \tag{6}$$

where i is the joint 1, 2, and 3, f is the frequency of the sinusoidal type of motion, and it is variable and linearly increasing.

This type of kinematic is similar to the motion presented in our previous work [46]. The range of each joint motion is $[-90, 90]\, deg$. The position and the orientation of the robot end-effector is calculated according to Equation (1) considering the following parameters: $L_1 = 0.31\, m$, $L_2 = 0.40\, m$, and $L_3 = 0.47\, m$. The number of the collected samples is 30395. All these data are presented from Fig. 4, Fig. 5, and Fig. 6.

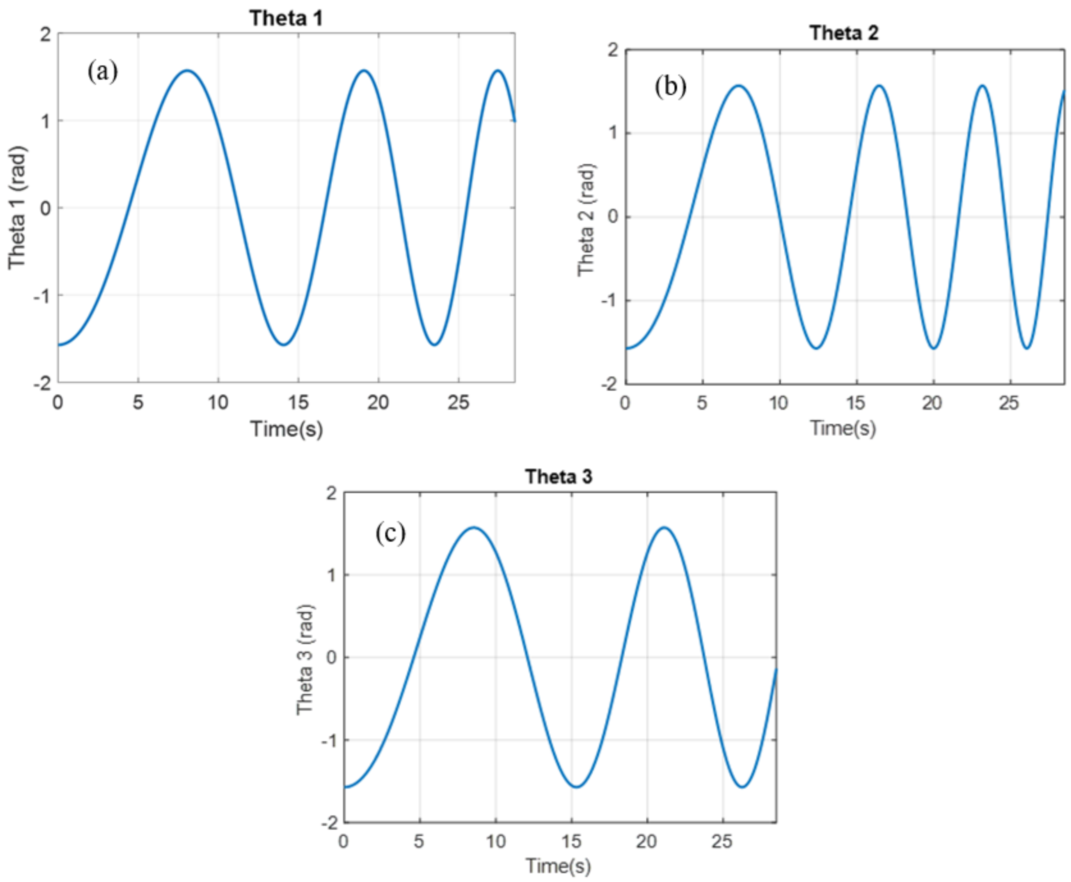


Fig. 4. The joints' variables θ_1, θ_2, and θ_3 in radians

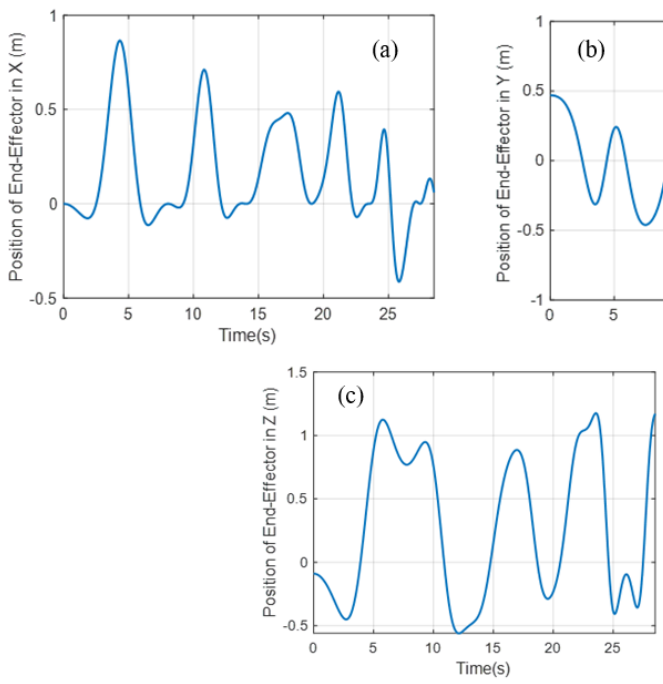


Fig. 5. The position of the robot end-effector p_x, p_y, and p_z in meters

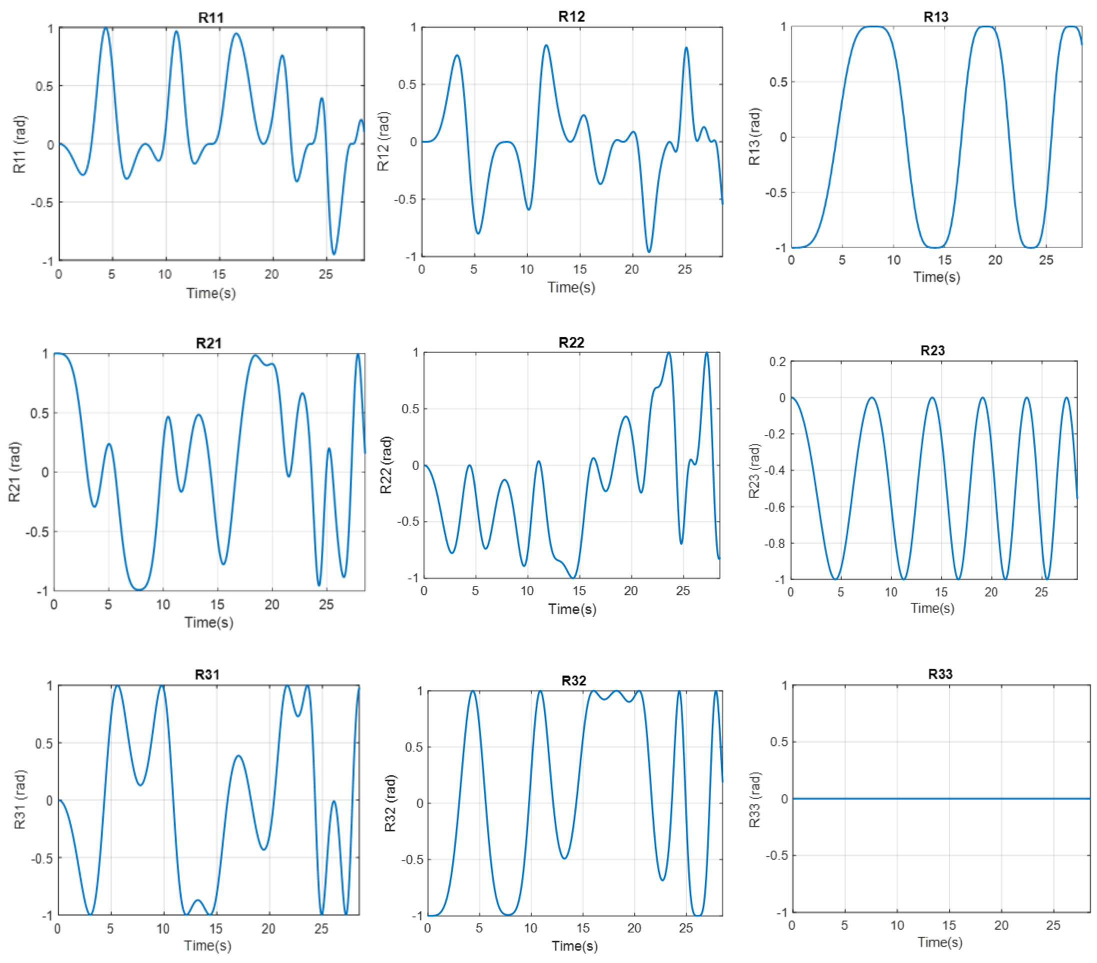


Fig. 6. The orientations of the robot's end-effector R_{11}, R_{12}, R_{13},……. R_{33} in radians

4. Forward Kinematics Solution Using MLFFNN

This section presents designing, training, testing, and verifying the proposed MLFFNN to solve the forward kinematics of a 3-DOF articulated manipulator.

4.1. Network Design

The proposed MLFFNN is designed using three layers. The first layer is the input layer which contains the three inputs, which are the joints' variables θ_1, θ_2, and θ_3. The hyperbolic tangent, denoted by tanh, serves as the activation function for the second layer, which is the hidden, non-linear layer. It also contains hidden neurons. The output layer is the third layer. This layer is linear and estimates the values of the position and the orientations for the targeted robot (p'_x, p'_y, p'_z, R'_{11}, R'_{12}, R'_{13}……R'_{33}), which are twelve outputs. These estimated outputs are compared with the actual positions and orientations (which are presented in Fig. 5 and Fig. 6). The main and followed criteria during the design of the MLFFNN-network is implementing a simple NN can achieve high levels of performance which can be represented by small mean squared error and training error. In other meaning, these values should be close to zero value. The proposed MLFFNN is presented in Fig. 7.

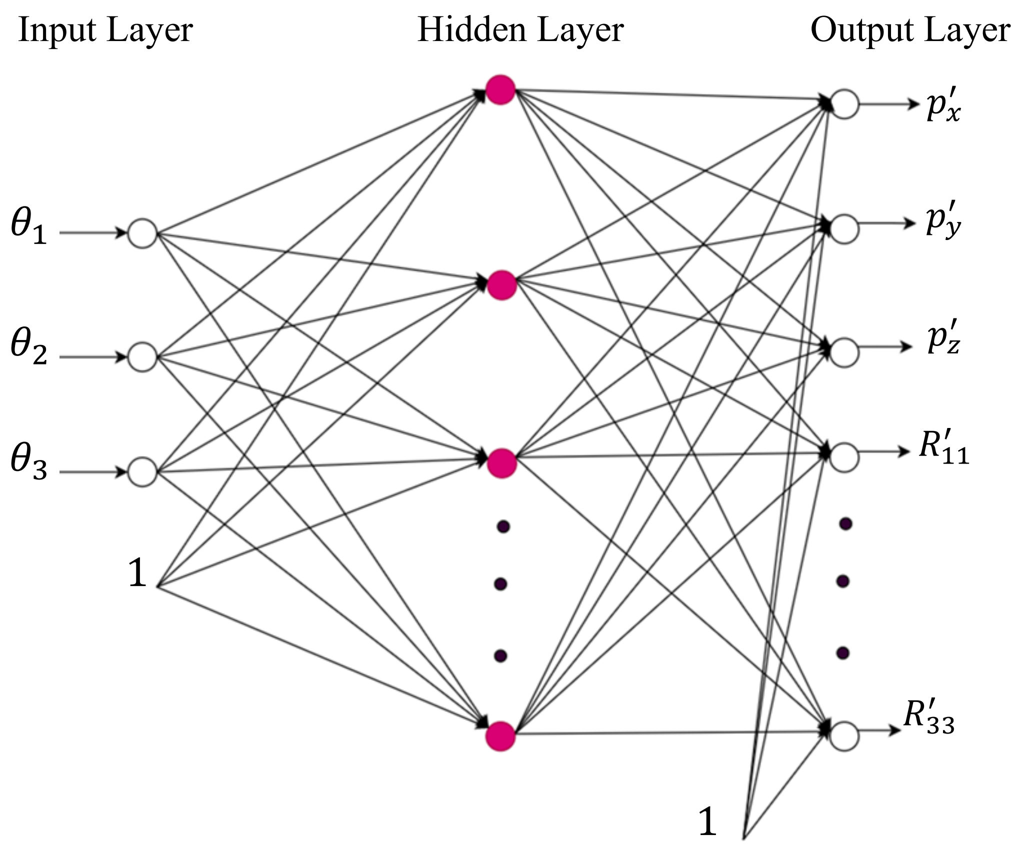


Fig. 7. The MLFFNN was used as a solution for the kinematics of the 3-DOF articulated manipulator in a forward mode. The inputs are the joints' variables, and the outputs are the positions and the orientations of the end-effector for the robot. The hidden layer includes 40 hidden neurons. The figure is drawn using an online program available at https://app.diagrams.net/

The intended end-effector positions and orientations of the robot are used only for the training of the designed MLFFNN. In addition, the error resulting from training must be small as possible and close to zero. The training process is described in the following subsection.

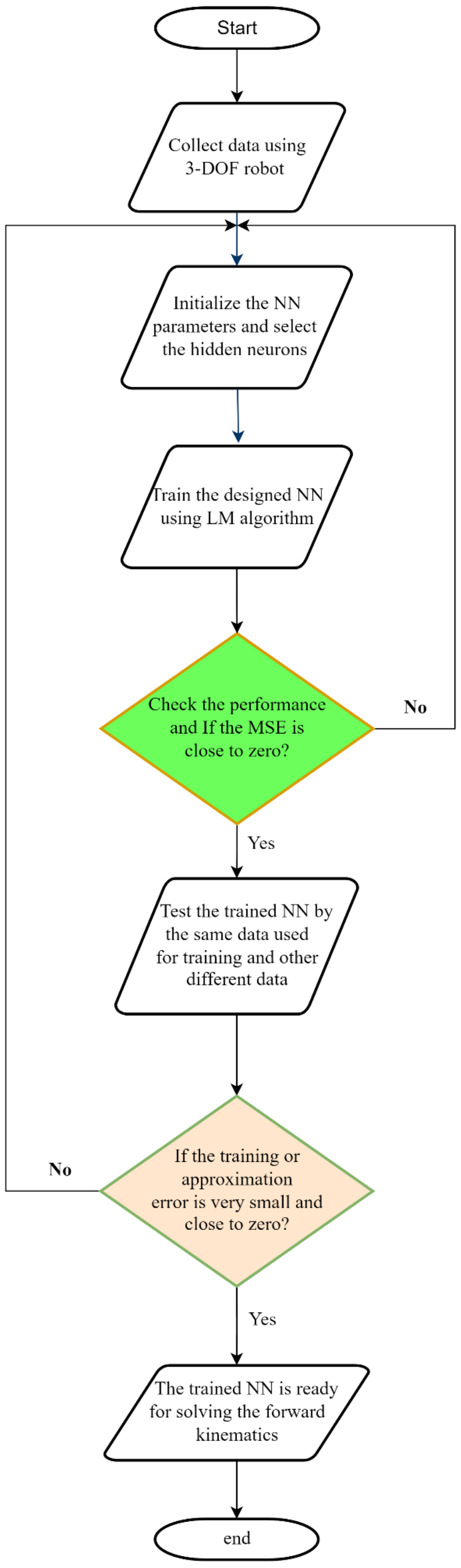

Start
Collect data using 3-DOF robot
Initialize the NN parameters and select the hidden neurons
Train the designed NN using LM algorithm
Check the performance and If the MSE is close to zero?
No
Yes
Test the trained NN by the same data used for training and other different data
If the training or approximation error is very small and close to zero?
No
Yes
The trained NN is ready for solving the forward kinematics
end

Fig. 8. The following steps for the train and the test processes of the MLFFNN in solving the forward kinematics. This graphical chart was sketched by the available online program at https://app.diagrams.net/

4.2. Designed Network Training

The collected data that are presented in section 3 are used for analyzing the behavior of the proposed MLFFNN network. These data are divided into three groups as follows:

1) 80% of these data (24315 samples) are used for the training,
2) 10% (3040 samples) are used for validation, and
3) The last 10% (3040 samples) are used for testing.

This division in data occurs in MATLAB randomly. In addition, this is very important to be able to test and validate the trained NN by different types of data used for the training procedure. Therefore, we can be sure that the trained NN works effectively and can estimate the output value correctly. The training is occurring in MATLAB using LM learning. The steps and methodology that followed during the training and test processes of the designed MLFFNN network are presented in Fig. 8.

After executing many experiments and trials by using different initialization of weights and different hidden neurons number, the preferred settings of parameters that enable the MLFFNN's high performance are obtained. These parameters are the best number of hidden neurons, the number of the used epochs or iterations, the very small MSE value, and the training time. All these parameters are presented in Table 2. Some other resulting parameters are presented in the Appendix. The intended high performance is achieving the lowest MSE as well as the training error. The time of training is not a very important issue because the training happens offline, and the main aim is to have a very well-trained MLFFNN network that can estimate the outputs efficiently with an error close to zero. The MSE value is calculated using Equation (7),

$$MSE = \sum_{i=1}^{n} \frac{(ActOut(i) - EstOut(i))^2}{n} \tag{7}$$

where $ActOut$ is the actual output that is used for the training process, and $EstOut$ is the estimated output by the designed NN. n is the number of samples.

Table 2. Obtained best parameters which lead to the high performance of the designed MLFFNN for forward kinematics solution

Parameter	Value
Number of hidden neurons	40
Number of epochs (iterations)	1000
Lowest MSE	$\mathbf{4.592 \times 10^{-8} \cong 0}$
Training time	1 hour, 33 minutes, and 1 second

The MSE and the regression obtained from the training are shown in Fig. 9 and Fig. 10. The resulting error histogram is presented in the Appendix. As presented in Fig. 9, the obtained MSE is a very small value and approximately zero. In addition, the resulting regression is 1, as shown in Fig. 10. These results illustrate that the required positions and orientations for the targeted robot end effector converge/coincide with the corresponding estimated ones by the MLFFNN network. In other meaning, the training error between them is approximately zero. The simulation results demonstrate that the analyzed MLFFNN network is learned and trained efficiently, which can estimate the positions and the orientations of the robot’s end-effector. The process of testing and the verification of the trained method is presented in the following subsection. This process of very important to show the effectiveness and reliability of the method.

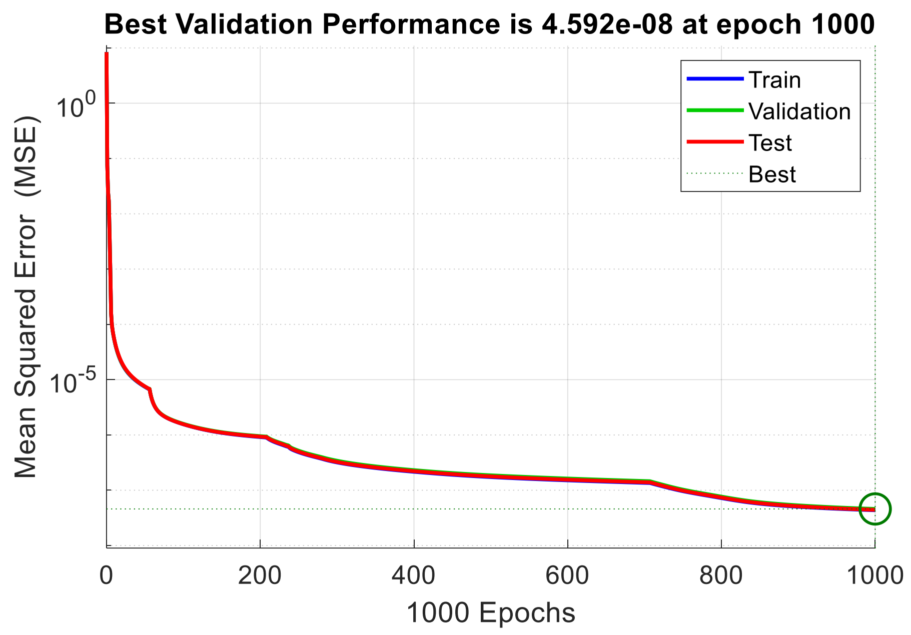


Fig. 9. The gained lowest MSE from training the designed MLFFNN for forward kinematics

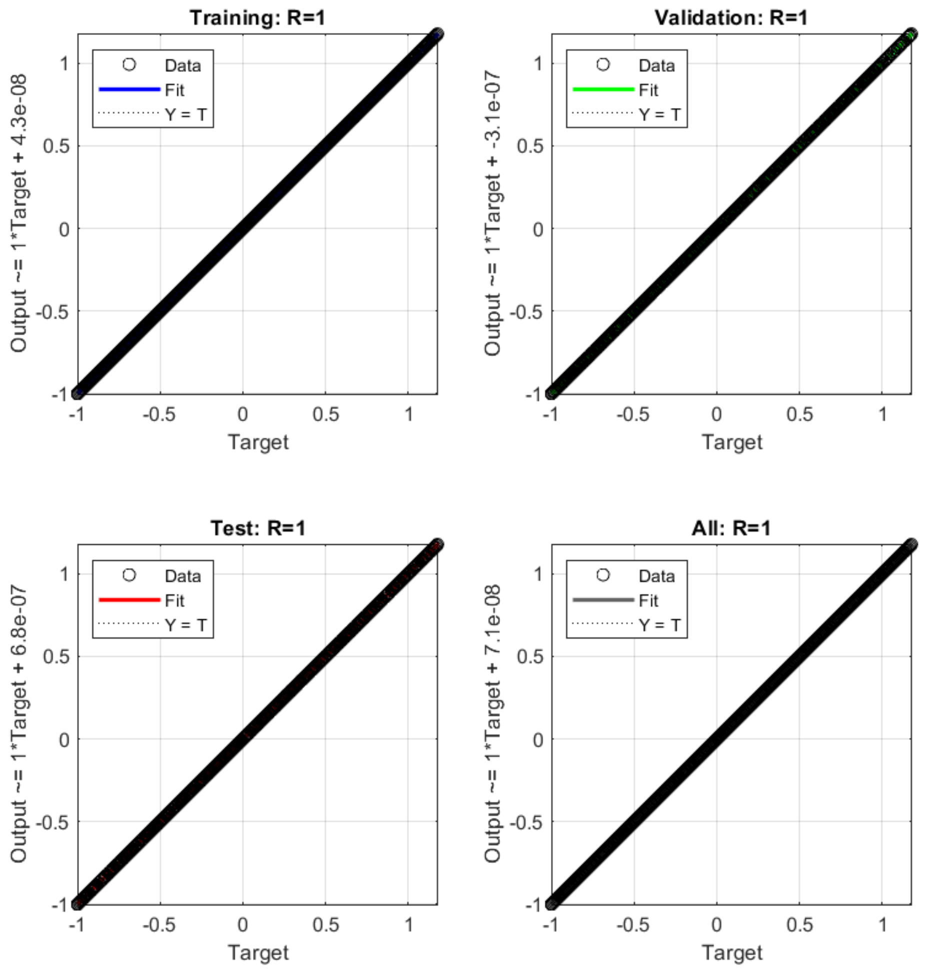


Fig. 10. The obtained regression from training the designed MLFFNN for forward kinematics

4.3. Trained Network Verification and Testing

All samples (30395) are used to verify and test the trained MLFFNN. Results and comparisons of the robot's desired effector's positions and orientations, and the corresponding estimated ones by the trained network are presented from Fig. 11 to Fig. 14. Furthermore, the average, the maximum, the minimum, and the standard deviation (std) of the approximation error between the desired and estimated end-effector positions are shown in Table 3.

From Fig. 11, Fig. 12, Fig. 13 and Table 3, it is clear that desired positions of the robot end-effector in $x -$, $y -$, and $z -$ directions and the corresponding estimated ones by the NN are coincide. The approximation error between them is a very small value and approximately zero. In Fig. 14, the desired orientations of the robot also coincide with the corresponding approximated ones by the neural network. This also means that the approximation error is a very small value and about the zero value. These results prove that the trained MLFFNN is highly reliable in minimizing errors and is trained very well. Furthermore, it is able to solve the forward kinematics (finding positions and orientations of end-effector) of a 3-DOF articulated manipulator in a correct way.

Table 3. The resulting parameters values: average, maximum, minimum, and standard deviation (std) of the approximation error between the desired and estimated end-effector positions

Parameter		**Position in $x -$direction**	**Position in $y -$ direction**	**Position in $z -$ direction**
Approximation of absolute error between the desired and estimated positions of the end-effector	**Average**	1.4058e-04	2.2879e-04	9.8047e-05
	Maximum	7.9550e-04	0.0036	7.1878e-04
	Minimum	4.5048e-09	2.6214e-09	6.1091e-10
	Standard deviation (std)	1.1052e-04	2.3766e-04	8.4661e-05

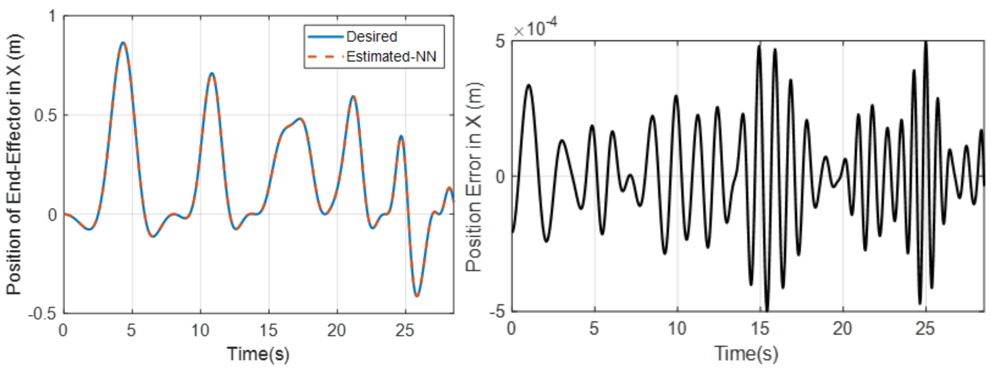


Fig. 11. The comparison between the desired and estimated position of the robot's end-effector in $x -$ direction. (a) p_x and p'_x, (b) $e_1(t) = p_x(t) - p'_x(t)$

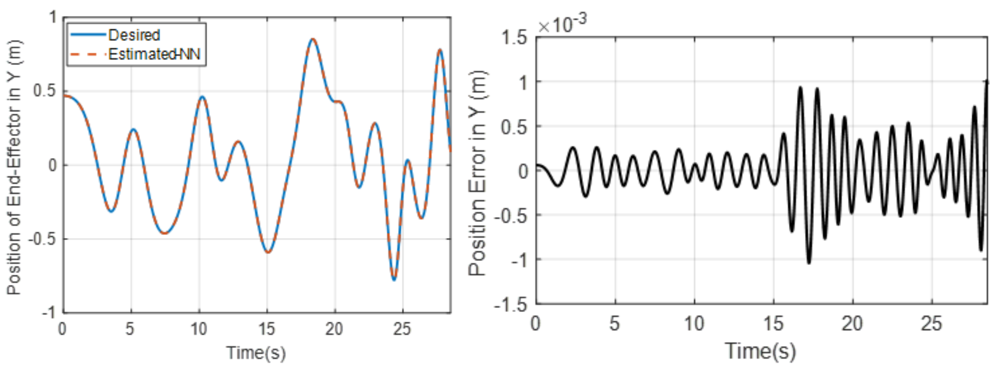


Fig. 12. The simulation comparative results between the desired and estimated position of the robot's end-effector in $y -$ direction. (a) p_y and p'_y, (b) $e_2(t) = p_y(t) - p'_y(t)$

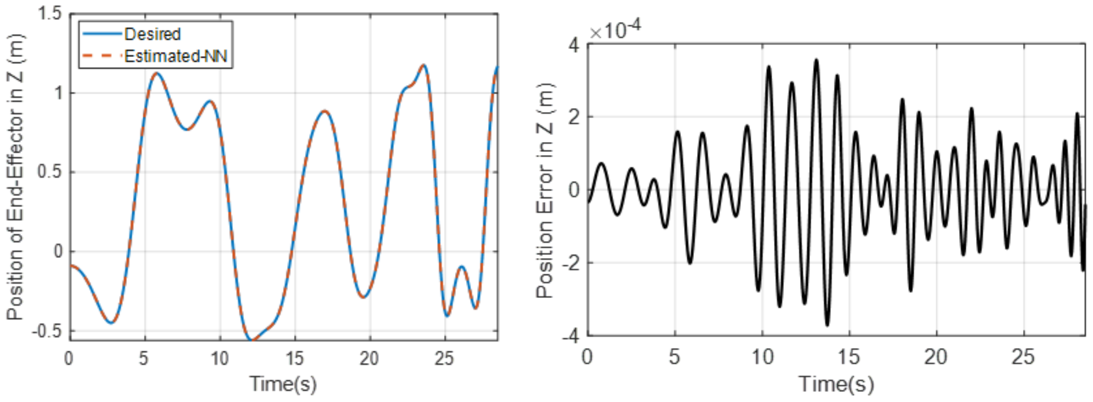


Fig. 13. The simulation comparative results between the desired and estimated position of the robot's end-effector in $z-$ direction. (a) p_z and p_z', (b) $e_3(t) = p_z(t) - p_z'(t)$

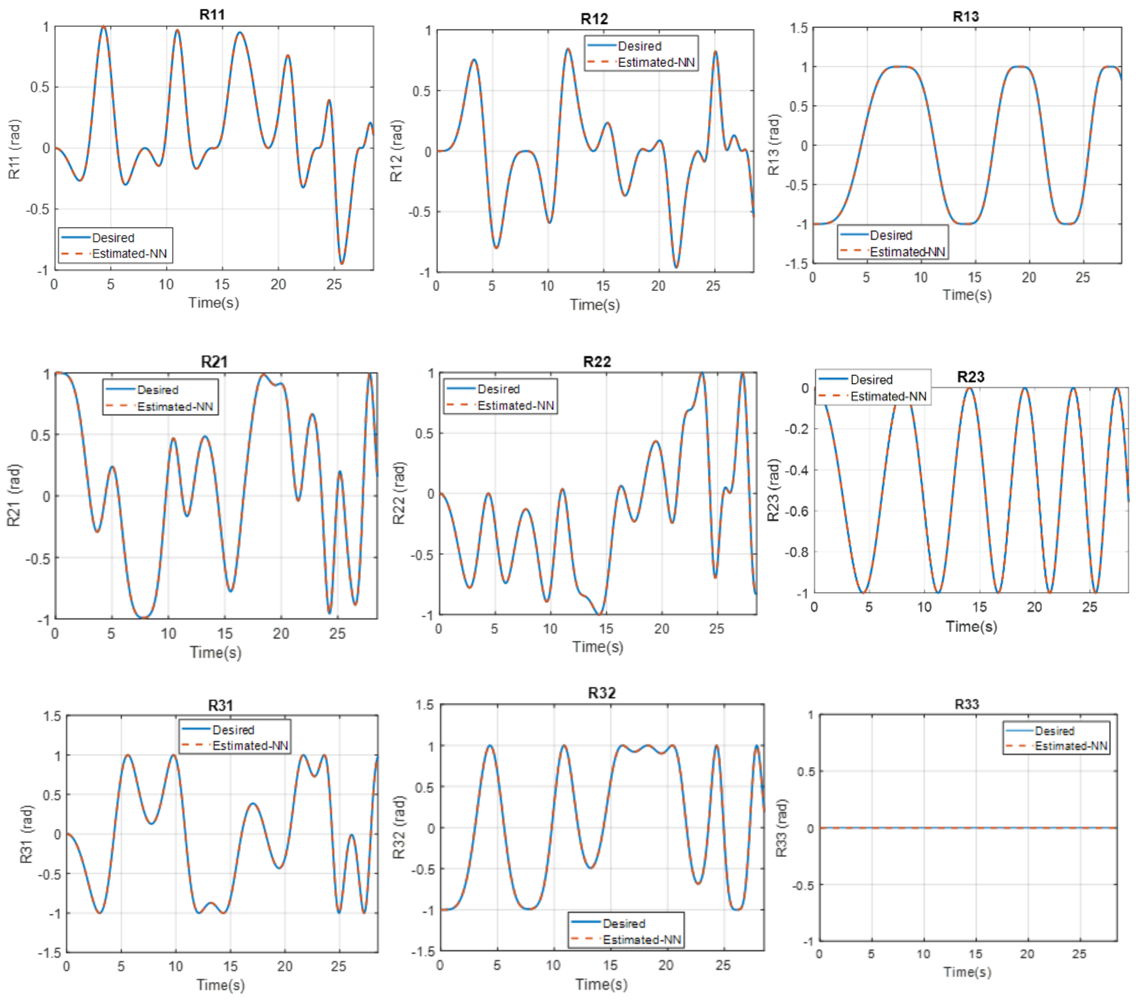


Fig. 14. The simulation comparative results between the desired orientations of the robot's end-effector and corresponding actual ones by the trained MLFFNN-network

It should be noted that the trained MLFFNN takes 1.379 seconds in MATLAB to do all the calculations and give the results. This time is very short, and if the method is applied experimentally/practically to any 3-DOF robot, it cannot affect the continuity of the robot' motion.

5. Inverse Kinematics Solution Using MLFFNN

This section shows in detail the designing, training, testing, and verifying of the proposed implemented MLFFNN to resolve the 3-DOF articulated robotic manipulator's inverse kinematics. Fig. 15 shows his design of MLFFNN.

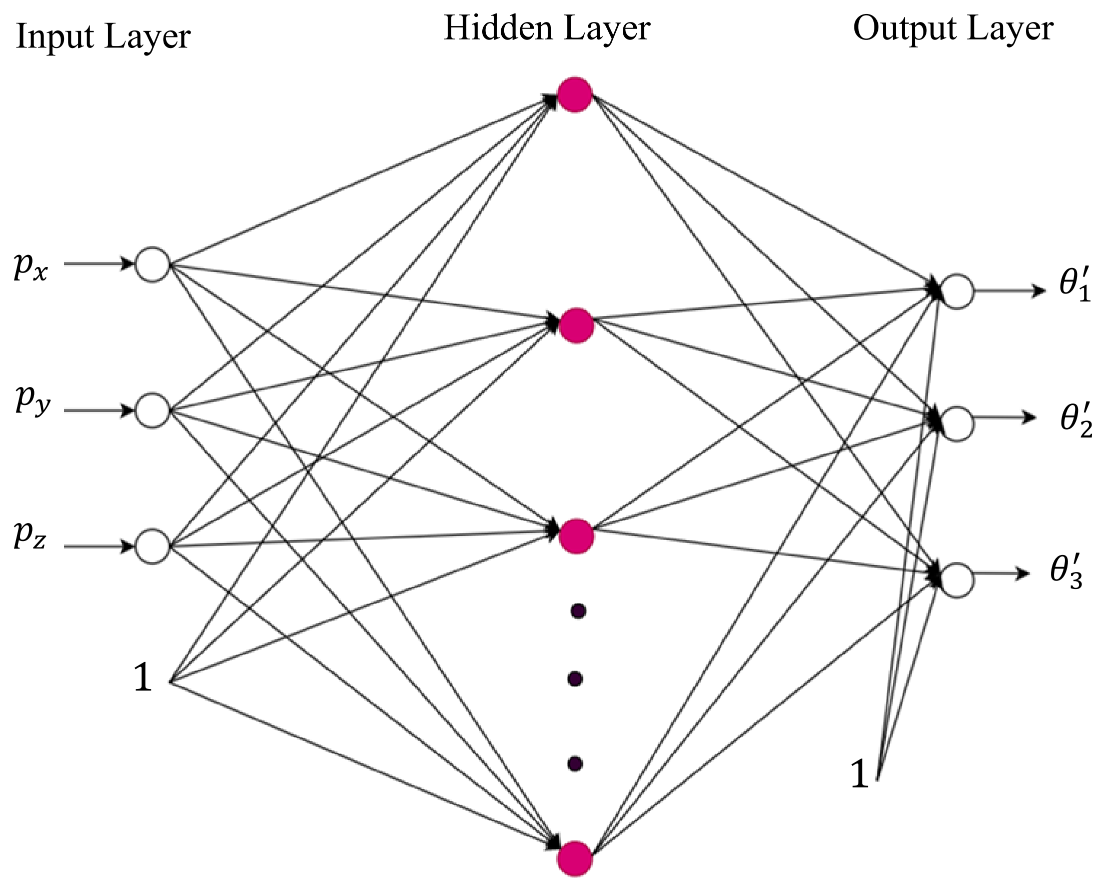


Fig. 15. The designed MLFFNN is used to solve the kinematics of the 3-DOF articulated manipulator in reverse mode. The inputs are the positions of the end-effector, and the outputs are the joints' variables. The hidden layer includes 120 hidden neurons. The figure is drawn using the online program available at https://app.diagrams.net/

5.1. Network Design

The proposed MLFFNN network is designed in such a way that using three layers of neurons are interconnected with each other. The first layer is called the input layer, which contains the three inputs, which are only the positions of the robot's end-effector (p_x, p_y, and p_z). The second layer, which contains hidden neurons, is called the hidden layer, which is considered non-linear in its operation, and the activation function for this layer is the hyperbolic tangent which is represented by tanh. The third and last layer is the output layer. This layer is linear and estimates the joints' variables (θ_1', θ_2', and θ_3'), which are three outputs. These estimated outputs are compared with the actual (desired) joints' variables (which are presented in Fig. 4). This proposed MLFFNN is presented in Fig. 15.

The desired joints' variables are utilized for training the designed MLFFNN network. The error from the training phase must be very small and close to the zero value. In detail, training for the implemented MLFFNN is described in the following subsection (5.2. Designed Network Training).

5.2. Designed Network Training

The same procedure that is presented in Subsection 4.2 is followed here. The collected data mentioned in Section 3 is divided as follows:

1) 80% of the collected data (24315 samples) are used for the training,
2) 10% (3040 samples) are used for validation, and
3) The last 10% (3040 samples) are used for testing.

The most suitable parameters which lead to high levels of performance for the designed MLFFNN network are the best number of hidden numbers, the number of epochs or iterations, the very small MSE, and the training time. These parameters are presented in Table 4. As mentioned in Section 4, the training time is not very important. Some other parameters are presented in the Appendix.

Table 4. The obtained best settings that were obtained led to the high performance of the designed MLFFNN for kinematics solution in an inverse method

Parameter	Value
Number of hidden neurons	120
Number of epochs (iterations)	1000
Lowest MSE	$9.071 \times 10^{-7} \cong 0$
Training time	44 minutes and 16 seconds

The resulting MSE and regression from the training are shown in Fig. 16 and Fig. 17. The obtained error histogram is presented in the Appendix. As shown from the figures, the MSE is very small and approximately zero. The regression is equal to 1. This means that the desired joints' variables are coinciding/converging with corresponding estimated ones by the implemented NN. Therefore, the training error is very small. This indicates that the designed MLFFNN-neural network is trained efficiently and it is ready to solve inverse kinematics correctly. Testing and the verification of the trained NN should be investigated, as presented in the following section. This process is very crucial for showing the effectiveness of the trained method.

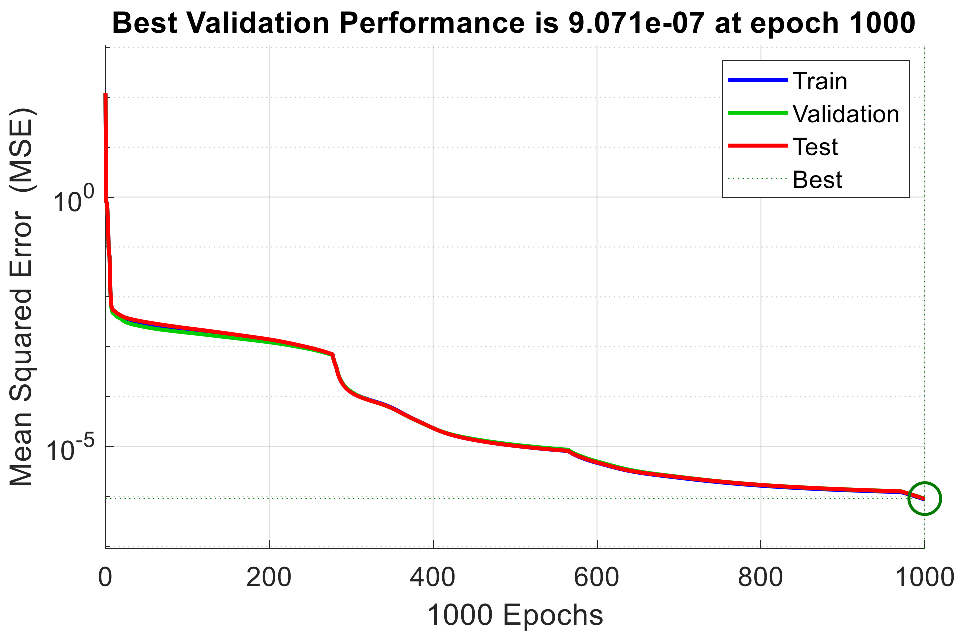


Fig. 16. The obtained lowest MSE from training the designed MLFFNN for inverse kinematics

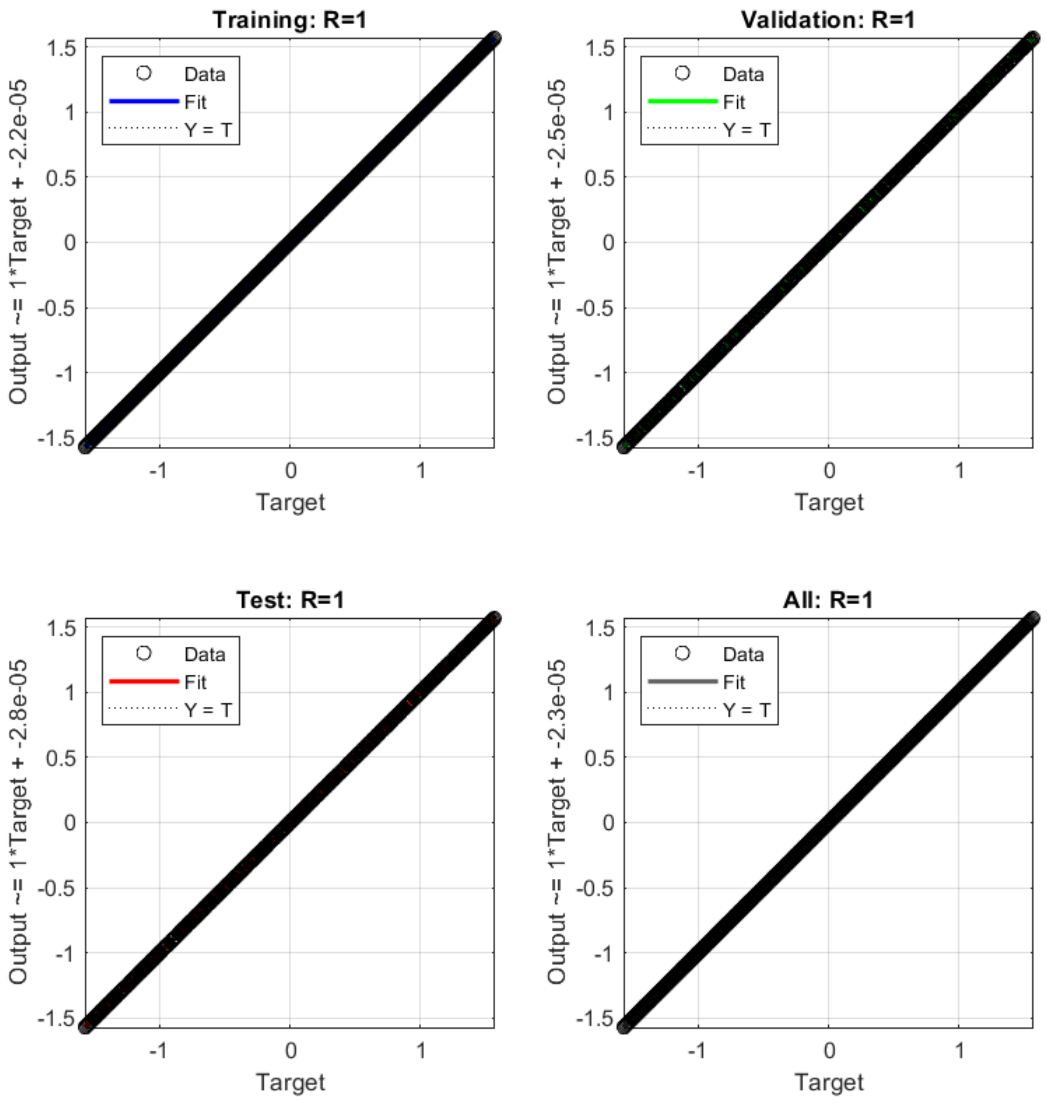


Fig. 17. The obtained regression from training the designed MLFFNN for inverse kinematics

5.3. Trained Network Verification and Testing

In this step, all samples (30395) are used to verify and test the trained MLFFNN. The results and the comparisons between the targeted joints' variables and the corresponding computed ones by the trained MLFFNN-neural network are presented in Fig. 18, Fig. 19, and Fig. 20. Furthermore, Table 5 presents the average, maximum, minimum, and standard deviation of the approximation absolute error between the desired and estimated joints' variables.

Table 5. The resulting parameters values: average, maximum, minimum, and standard deviation (std) of the approximation absolute error between the desired and estimated joints' variables

Parameter		Theta 1	Theta 2	Theta 3
Approximation of absolute error between the desired and estimated joints' variables	**Average**	7.9624e-04	7.3061e-04	6.5449e-04
	Maximum	0.0043	0.0044	0.0032
	Minimum	2.8322e-08	4.7931e-08	1.5800e-08
	Standard deviation (std)	6.2181e-04	6.1428e-04	5.2187e-04

It is clear from the presented figures (Fig. 18, Fig. 19, and Fig. 20) and Table 5 that the desired joints' variables (Theta1, Theta 2, and Theta 3) coincide with the corresponding estimated ones by the

NN. The approximation error between them is a very small value and approximately zero value. This proves that the trained MLFFNN is highly reliable in minimizing error and is trained very well.

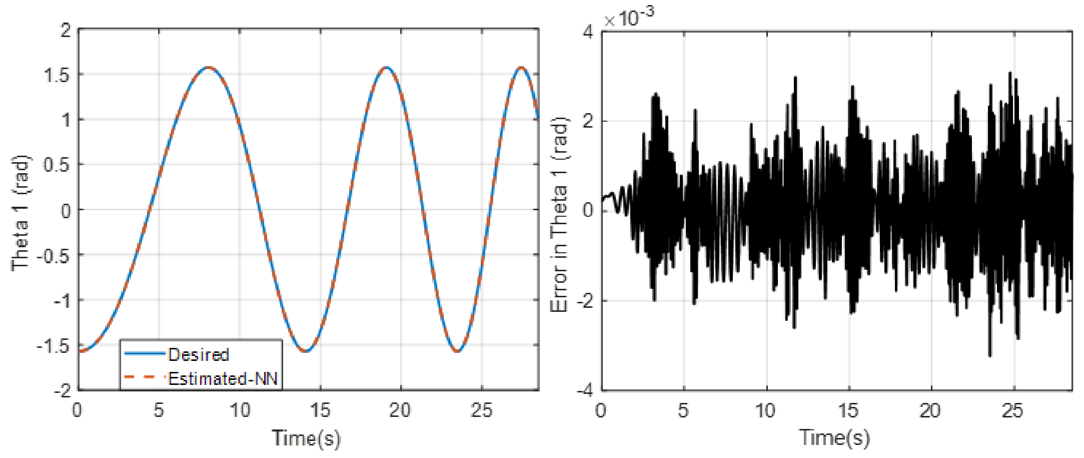


Fig. 18. The comparison between the desired and estimated joint variable Theta 1. (a) θ_1 and θ_1', (b) $e_1(t) = \theta_1(t) - \theta_1'(t)$

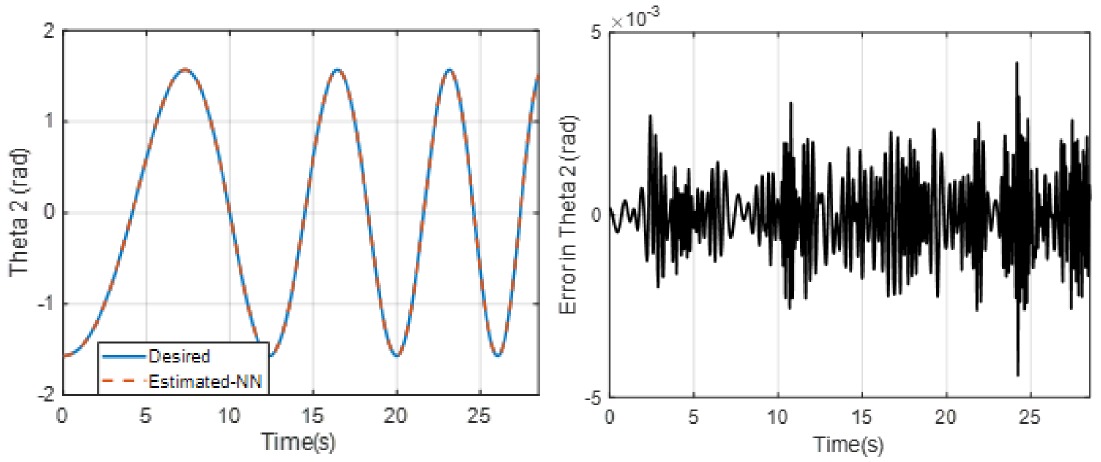


Fig. 19. The comparison between the desired and estimated joint variable Theta 2. (a) θ_2 and θ_2', (b) $e_2(t) = \theta_2(t) - \theta_2'(t)$

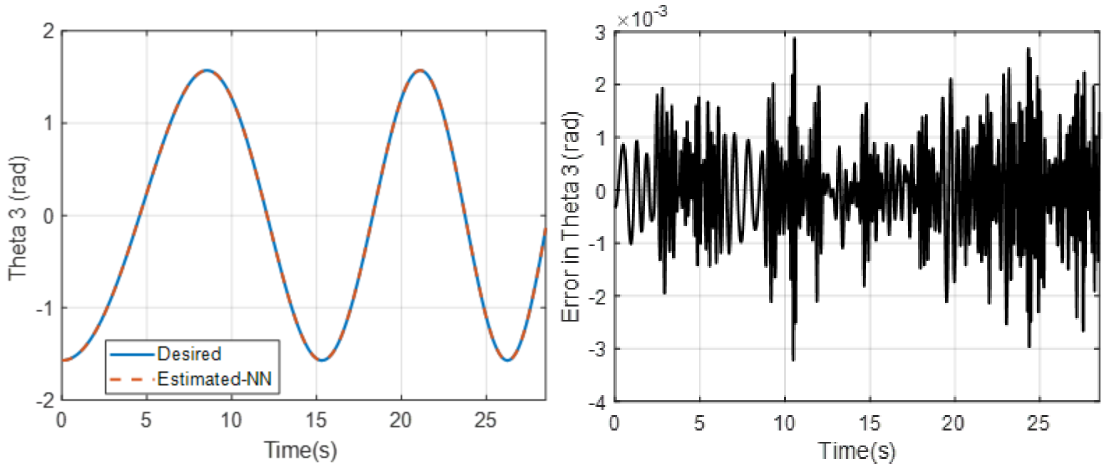


Fig. 20. The comparison between the desired and estimated joint variable Theta 3. (a) θ_3 and θ_3', (b) $e_3(t) = \theta_3(t) - \theta_3'(t)$

It should be noted that the trained MLFFNN takes 0.587 seconds in MATLAB to do all the calculations and give the results. This time is very short, and if the method is applied experimentally/practically to any 3-DOF robot, it cannot affect the continuity of the robot' motion. The time in the case of the inverse kinematics is less than the time in the forward kinematics due to the fact that MLFFNN-network estimates 12 outputs in the forward kinematics case, and it estimates only 3 outputs in the other case.

Remark: To solve the inverse kinematics, we created another MLFFNN and used both the positions and orientations of the robot end-effector as its inputs. The results are better compared with the presented one in this section (Section 5). However, the presented MLFFNN in this section (Section 5) gives very good results because the MSE and training/approximation error are both very low values that are close to zero. In addition, we prefer it since the inputs are smaller than when using the end-effector locations and orientations. The complexity of the presented MLFFNN is also lower.

Limitation of the proposed method: The proposed method is investigated using a limited range of the joints' motion of the manipulator as the range of each joint motion is $[-90, 90]\ deg$. Therefore, applying the method using the entire experiment space of the robot joints should be investigated and considered in future work. The proposed method is used with a 3-DOF articulated robot. For investigating the generalization of the proposed network, it should be applied to other different types of robots and more complex robots like 6-DOF robots or 7-DOF robots.

In the current work, only sinusoidal motion is recommended for the robot joints. Other different types of joint motion should also be considered and used. The proposed method is validated only using simulated data, and no experimental data or real robot experiments are conducted to validate the effectiveness of the proposed method in real-world scenarios. This happens because we do not have a real robot at the time of doing this paper. However, in future work, the experimental validation should be considered and investigated.

The accuracy of the proposed MLFFNN for solving the problem of a 3-DOF manipulator working in the inverse kinematics is compared with other previous NNs-based approaches such as the ones presented by Koker et al. [47], Daya et al. [48], Duka [3], and Jiménez-López et al. [49]. With Duka [3], a feedforward NN was used for solving the inverse kinematics of a 3-DOF planar manipulator. The NN was designed using six inputs which were the three end-effector positions and the three orientations. With Koker et al. [47], a backpropagation NN with a sigmoidal activation function was proposed as a foundation for the inverse motion problem for the 3-DOF manipulator robot. In [48], Daya et al. developed an NN-based approach for the inverse kinematics solution of a 3-DOF manipulator.

In their design, six inputs were used, which were the three positions and the three orientations of the end-effector. Jiménez-López et al. [49] implemented a NN for the inverse kinematic solution of a 3-DOF manipulator. In their approach, the NN was trained using Bayesian regularization backpropagation. In addition, the limited range of the joints' motion of the manipulator was considered. The comparison includes the resulting MSE, which is the main parameter to show the accuracy of each method. The comparison is shown in Fig. 21. As shown in Fig. 21, the proposed method has the resulting smallest MSE value compared with other previous related approaches. Hence, the results prove that the proposed method is the most accurate, and the resulting approximation error is the smallest.

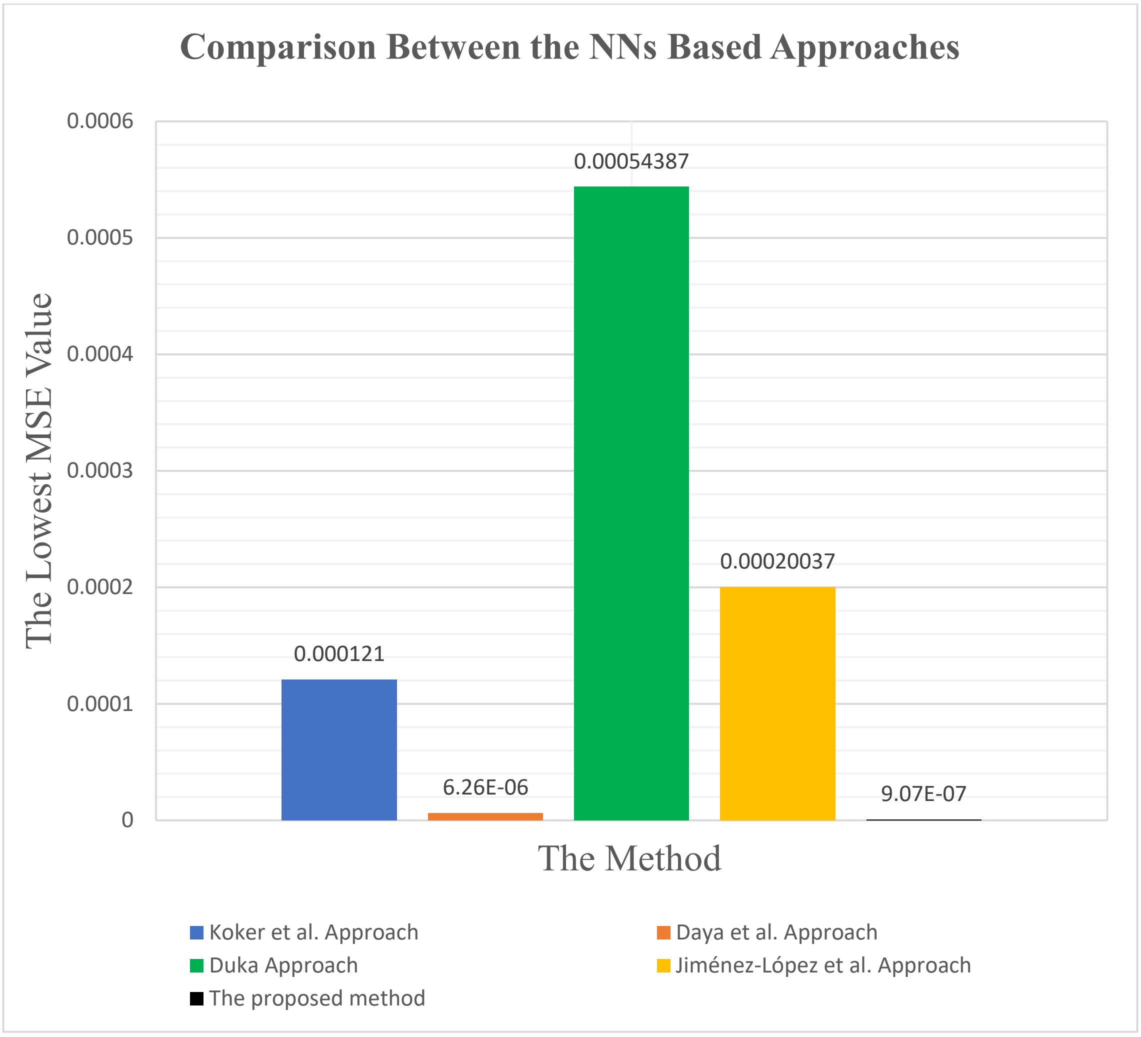


Fig. 21. The comparison between the proposed method for solving the inverse kinematics of a 3-DOF manipulator and other previous related approaches

6. Conclusion

In this research, the kinematics of a 3-DOF articulated robotic manipulator in both forward and inverse modes are solved using the concept of the MLFFNN-network algorithm. For Forward kinematics, the MLFFNN is designed using the joints' variables as its inputs. In addition, its outputs are the positions and orientations of the end-effector of the robot. For the kinematics in the inverse case, only the positions of the end-effector are used as the inputs of the MLFFNN. The outputs are the joints' variables. For both cases, the designed MLFFNN network is trained using the LM learning algorithm using data collected from the sinusoidal joint motion of the manipulator. The resulting training errors and the MSE from the training stage have relatively very small values and close to zero values. The simulation results demonstrate that the proposed neural network is trained in an effective way. The trained MLFFNN is tested and verified, and the results prove that the approximation error between the actual output and the corresponding predicted one by the NN is equal to a very small value. Therefore, the MLFFNN is highly reliable in minimizing errors. Furthermore, it is efficient to solve the kinematics of the robot in forward and inverse modes correctly. Our proposed research approach is compared with other previously published methods. The experimental simulation result reveals that our proposed neural network-based method has the highest levels of accuracy compared with others.

7. Future Work

Different aspects of future work can be discussed in this section. Firstly, as the range of each joint motion of the used robot is limited in the current work and is $[-90, 90]\ deg$, the future work can consider the whole workspace of the robot joints. Secondly, the use of the NN in solving the forward and inverse kinematics of a complex robot, such as a 6-DOF or a 7-DOF manipulator, can also be considered. Thirdly, different types of NNs-based architectures inspired by biological concepts can be used and compared, such as WaveNet, RNN, and RBF. Deep learning methods can also be investigated and compared. Fourthly, we would synthesize the hardware implementation for the proposed MLFFNN network on file programmable gate array (FPGA) technology and use different fault-tolerant techniques. FPGA has many advantages, as follows:

1) It can easily change its functionality after designing the product.
2) It does not require a larger board area, and it is more energy efficient than other equivalent discrete circuits.
3) It can carry out several operations on data simultaneously.

Finally, investigating the current work experimentally with a real robot is recommended.

Abbreviations

Abbreviation	Meaning
DH	Denavit–Hartenberg
POE	Product of Exponential
SVR	Support Vector Regression
NN	Neural Network
DOF	Degree of Freedom
LWR	Light Weight Robot
MLFFNN	Multilayer Feedforward Neural Network
LM	Levenberg–Marquardt
MSE	Mean Squared Error
Tanh	Hyperbolic Tangent
FPGA	File Programmable Gate Array

Supplementary Materials: The generated datasets during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Author Contribution: All authors contributed to this paper. Most of the presented work is done by Abdel-Nasser Sharkawy. All authors read and approved the final paper.

Funding: This research received no external funding

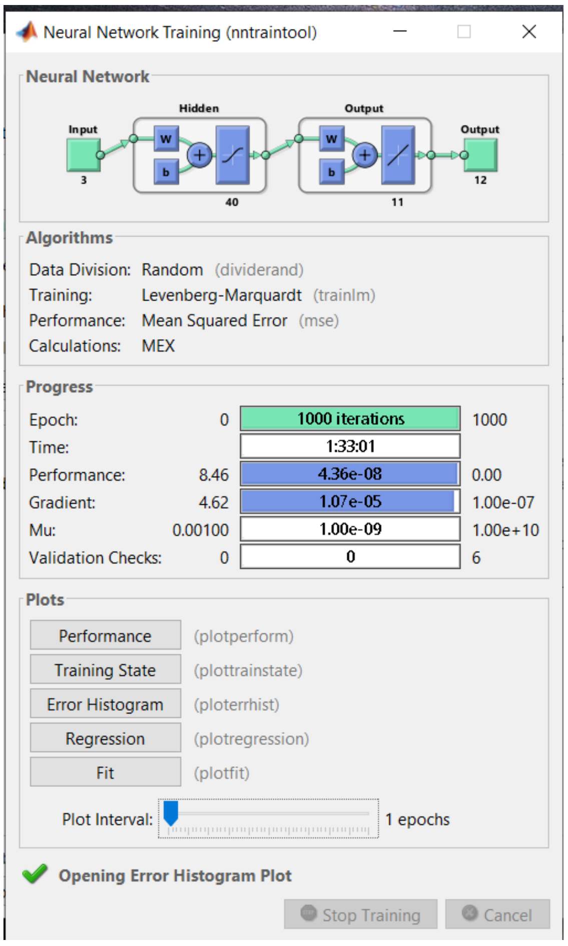
Acknowledgment: It is not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix

Part 1: Some parameters resulted from the training process of the MLFFNN that is used for solving the forward and inverse kinematics problem. These parameters are shown in Fig. A1.

Part 2: The obtained error histogram from the MLFFNN training in the forward and inverse kinematics cases. These histograms are presented in Fig. A2. As shown from these histograms, the error, which is the difference between the actual output and the estimated one by NN, is very close to zero. This supports that the proposed NN is trained in a very effective way.



Neural Network Training (nntraintool)

Neural Network

Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	1000 iterations	1000
Time:		1:33:01	
Performance:	8.46	4.36e-08	0.00
Gradient:	4.62	1.07e-05	1.00e-07
Mu:	0.00100	1.00e-09	1.00e+10
Validation Checks:	0	0	6

Plots

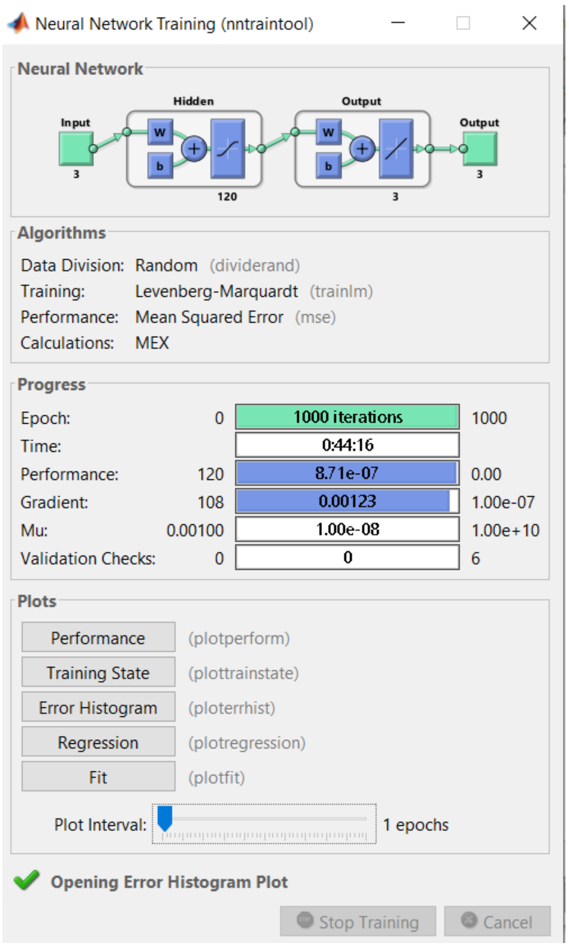
Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)

Plot Interval: 1 epochs

✓ Opening Error Histogram Plot

Stop Training Cancel

(a) In the forward kinematics case



Neural Network Training (nntraintool)

Neural Network

Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

Progress

Epoch:	0	1000 iterations	1000
Time:		0:44:16	
Performance:	120	8.71e-07	0.00
Gradient:	108	0.00123	1.00e-07
Mu:	0.00100	1.00e-08	1.00e+10
Validation Checks:	0	0	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)
Regression (plotregression)
Fit (plotfit)

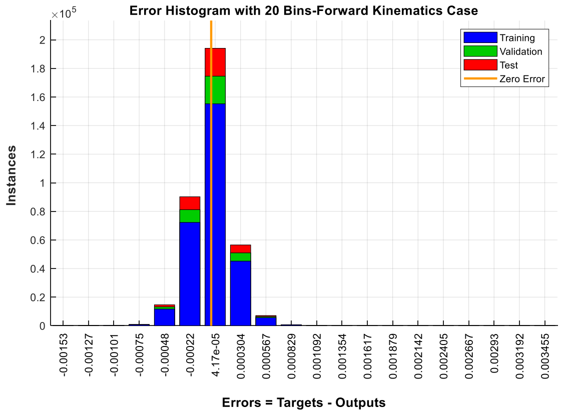
Plot Interval: 1 epochs

✓ Opening Error Histogram Plot

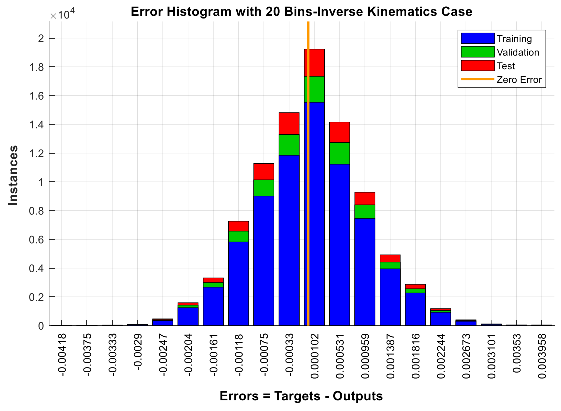
Stop Training Cancel

(b) In the inverse kinematics case

Fig. A1. Some parameters result from the training of the MLFFNN (a) The case that is used for the forward kinematics problem (b) The case that is used for the inverse kinematics problem



(a) The error histogram resulted in the case of the forward kinematics.



(b) The error histogram resulted in the case of inverse kinematics.

Fig. A2. The resulting error histogram from the training stage

References

[1] S. Kucuk and Z. Bingul, “Robot Kinematics: Forward and Inverse Kinematics,” in *Industrial Robotics: Theory, Modelling and Control*, p. 964, 2006, https://www.intechopen.com/chapters/379.

[2] R. Singh, V. Kukshal, and V. S. Yadav, “A review on forward and inverse kinematics of classical serial manipulators,” in *Lecture Notes in Mechanical Engineering*, pp. 417–428, 2021, https://doi.org/10.1007/978-981-33-4018-3_39.

[3] A. -V. Duka, “Neural Network based Inverse Kinematics Solution for Trajectory Tracking of a Robotic Arm,” *Procedia Technol.*, vol. 12, pp. 20–27, 2014, https://doi.org/10.1016/j.protcy.2013.12.451.

[4] F. Piltan, A. Taghizadegan, and N. B. Sulaiman, “Modeling and Control of Four Degrees of Freedom Surgical Robot Manipulator Using MATLAB/SIMULINK,” *Int. J. Hybrid Inf. Technol.*, vol. 8, no. 11, pp. 47–78, 2015, https://doi.org/10.14257/ijhit.2015.8.11.05.

[5] J. Denavit and R. S. Hartenberg, “A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices,” *J. Appl. Mech. Am. Soc. Mech. Eng.*, vol. 22, no. 2, pp. 215–221, 1955, https://doi.org/10.1115/1.4011045.

[6] M. Himanth and L. V. Bharath, “Forward Kinematics Analysis of Robot Manipulator Using Different Screw Operators,” *Int. J. Robot. Autom.*, vol. 3, no. 2, pp. 21–28, 2018, https://mechanical.journalspub.info/index.php?journal=IJoA&page=article&op=view&path%5B%5D=414.

[7] S. Asif and P. Webb, “Kinematics Analysis of 6-DoF Articulated Robot with Spherical Wrist,” *Math. Probl. Eng.*, vol. 2021, pp. 1–11, 2021, https://doi.org/10.1155/2021/6647035.

[8] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994, https://doi.org/10.1201/9781315136370.

[9] J. -H. Kim and V. R. Kumar, “Kinematics of robot manipulators via line transformations,” *J. Robot. Syst.*, vol. 7, no. 4, pp. 649–674, 1990, https://doi.org/10.1002/rob.4620070408.

[10] A. -N. Sharkawy and N. Aspragathos, "A comparative study of two methods for forward kinematics and Jacobian matrix determination," *2017 International Conference on Mechanical, System and Control Engineering (ICMSC)*, pp. 179-183, 2017, https://doi.org/10.1109/ICMSC.2017.7959467.

[11] J. S. Kim, Y. H. Jeong, and J. H. Park, “A geometric approach for forward kinematics analysis of a 3-SPS/S redundant motion manipulator with an extra sensor using conformal geometric algebra,” *Meccanica*, vol. 51, no. 10, pp. 2289–2304, 2016, https://doi.org/10.1007/s11012-016-0369-3.

[12] K. R. Müller, A. J. Smoła, G. Rätsch, B. Schölkopf, J. Kohlmorgen, and V. Vapnik, “Predicting time series with support vector machines,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1327, pp. 999–1004, 1997, https://doi.org/10.1007/BFb0020283.

[13] S. Haykin, *Neural Networks and Learning Machines*, Third Edit. Pearson, 2009, https://cours.etsmtl.ca/sys843/REFS/Books/ebook_Haykin09.pdf.

[14] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015, http://neuralnetworksanddeeplearning.com/.

[15] A. -N. Sharkawy, “Forward and inverse kinematics solution of a robotic manipulator using a multilayer feedforward neural network,” *J. Mech. Energy Eng.*, vol. 6, no. 2, pp. 1–17, 2022, https://doi.org/10.30464/jmee.00300.

[16] A. Morell, M. Tarokh, and L. Acosta, “Solving the forward kinematics problem in parallel robots using Support Vector Regression,” *Eng. Appl. Artif. Intell.*, vol. 26, no. 7, pp. 1698–1706, 2013, https://doi.org/10.1016/j.engappai.2013.03.011.

[17] H. Sadjadian and H. Taghirad, “Comparison of different methods for computing the forward kinematics of a redundant parallel manipulator,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 44, no. 3, pp. 225–246, 2005, https://doi.org/10.1007/s10846-005-9006-4.

[18] A. Ghasemi, M. Eghtesad, and M. Farid, “Neural network solution for forward kinematics problem of cable robots,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 60, no. 2, pp. 201–215, 2010, https://doi.org/10.1007/s10846-010-9421-z.

[19] A. A. Canutescu and R. L. Dunbrack, “Cyclic coordinate descent: A robotics algorithm for protein loop closure,” *Protein Sci.*, vol. 12, no. 5, pp. 963–972, 2003, https://doi.org/10.1110/ps.0242703.

[20] Y. Lou, P. Quan, H. Lin, D. Wei, and S. Di, “A closed-form solution for the inverse kinematics of the 2n-dof hyper-redundant manipulator based on general spherical joint,” *Appl. Sci.*, vol. 11, no. 3, pp. 1–19, 2021, https://doi.org/10.3390/app11031277.

[21] K. Waldron and J. S. Schmiedeler, “Chapter 1: Kinematics,” *Handbook of Robotics*, pp. 9–33, 2008, https://doi.org/10.1007/978-3-540-30301-5_2.

[22] A. O. Elnady, “Iterative Technique for Solving the Inverse Kinematics Problem of Serial Manipulator,” *J. Mech. Eng. Autom.*, vol. 10, no. 1, pp. 12–18, 2021, http://article.sapub.org/10.5923.j.jmea.20211001.02.html.

[23] P. Kalra, P. B. Mahapatra, and D. K. Aggarwal, “An evolutionary approach for solving the multimodal inverse kinematics problem of industrial robots,” *Mech. Mach. Theory*, vol. 41, no. 10, pp. 1213–1229, 2006, https://doi.org/10.1016/j.mechmachtheory.2005.11.005.

[24] R. Köker, “A neuro-genetic approach to the inverse kinematics solution of robotic manipulators,” *Sci. Res. Essays*, vol. 6, no. 13, pp. 2784–2794, 2011, https://academicjournals.org/journal/SRE/article-abstract/8364A8623219.

[25] S. Tejomurtula and S. Kak, “Inverse kinematics in robotics using neural networks,” *Inf. Sci. (Ny).*, vol. 116, no. 2, pp. 147–164, 1999, https://doi.org/10.1016/S0020-0255(98)10098-1.

[26] 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),” *Comput. Intell. Neurosci.*, vol. 2016, pp. 1–10, 2016, https://doi.org/10.1155/2016/5720163.

[27] A. Csiszar, J. Eilers and A. Verl, "On solving the inverse kinematics problem using neural networks," *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pp. 1-6, 2017, https://doi.org/10.1109/M2VIP.2017.8211457.

[28] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*, 2nd Editio. JOHN WILEY & SONS, INC., 2020, https://www.wiley.com/en-us/Robot+Modeling+and+Control%2C+2nd+Edition-p-9781119524045.

[29] A. Ashagrie, A. O. Salau, and T. Weldcherkos, “Modeling and control of a 3-DOF articulated robotic manipulator using self-tuning fuzzy sliding mode controller,” *Cogent Eng.*, vol. 8, no. 1, pp. 1–33, 2021, https://doi.org/10.1080/23311916.2021.1950105.

[30] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015, https://doi.org/10.1016/j.neunet.2014.09.003.

[31] A. -N. Sharkawy, “Principle of Neural Network and Its Main Types : Review,” *J. Adv. Appl. Comput. Math.*, vol. 7, pp. 8–19, 2020, https://doi.org/10.15377/2409-5761.2020.07.2.

[32] A. B. Rad, T. W. Bui, V. Li, and Y. K. Wong, “A New On-Line PID Tuning Method Using Neural Networks,” *IFAC Proceedings Volumes*, vol. 33, no. 4, pp. 443–448, 2000, https://doi.org/10.1016/S1474-6670(17)38283-6.

[33] S. A. Elbelady, H. E. Fawaz, and A. M. A. Aziz, “Online Self Tuning PID Control Using Neural Network for Tracking Control of a Pneumatic Cylinder Using Pulse Width Modulation Piloted Digital Valves,” *Int. J. Mech. Mechatronics Eng. IJMME-IJENS*, vol. 16, no. 3, pp. 123–136, 2016, http://ijens.org/Vol_16_I_03/163603-9898-IJMME-IJENS.pdf.

[34] R. H.- Alvarado, L. G. G.- Valdovinos, T. S.- Jiménez, A. G.- Espinosa, and F. F.- Navarro, “Neural Network-Based Self-Tuning PID Control for Underwater Vehicles,” *Sensors*, vol. 16, no. 9: 1429, pp. 1–18, 2016, https://doi.org/10.3390/s16091429.

[35] S. C. Chen, S. W. Lin, T. Y. Tseng, and H. C. Lin, “Optimization of back-propagation network using simulated annealing approach,” in *2006 IEEE International Conference on Systems, Man and Cybernetics*, pp. 2819–2824, 2006, https://doi.org/10.1109/ICSMC.2006.385301.

[36] M. A. Sassi, M. J. D. Otis, and A. C.- Lecours, “Active stability observer using artificial neural network for intuitive physical human–robot interaction,” *Int. J. Adv. Robot. Syst.*, vol. 14, no. 4, pp. 1–16, 2017,

https://doi.org/10.1177/1729881417727326.

[37] E. De Momi, L. Kranendonk, M. Valenti, N. Enayati, and G. Ferrigno, “A Neural Network-Based Approach for Trajectory Planning in Robot–Human Handover Tasks,” *Front. Robot. AI*, vol. 3, pp. 1–10, 2016, https://doi.org/10.3389/frobt.2016.00034.

[38] A. -N. Sharkawy and A. A. Mostfa, “Neural Networks Design and Training for Safe Human-Robot Cooperation,” *Journal of King Saud University - Engineering Sciences*, vol. 34, no. 8, pp. 582-596, 2022, https://doi.org/10.1016/j.jksues.2021.02.004.

[39] A. -N. Sharkawy, P. N. Koustoumpardis, and N. Aspragathos, “A neural network-based approach for variable admittance control in human–robot cooperation: online adjustment of the virtual inertia,” *Intell. Serv. Robot.*, vol. 13, no. 4, pp. 495–519, 2020, https://doi.org/10.1007/s11370-020-00337-4.

[40] A. -N. Sharkawy, P. N. Koustoumpardis, and N. Aspragathos, “A recurrent neural network for variable admittance control in human – robot cooperation : simultaneously and online adjustment of the virtual damping and Inertia parameters,” *Int. J. Intell. Robot. Appl.*, vol. 4, no. 4, pp. 441–464, 2020, https://doi.org/10.1007/s41315-020-00154-z.

[41] P. Jeatrakul and K. W. Wong, "Comparing the performance of different neural networks for binary classification problems," *2009 Eighth International Symposium on Natural Language Processing*, pp. 111-115, 2009, https://doi.org/10.1109/SNLP.2009.5340935.

[42] D. Anderson and G. McNeill, “Artificial neural neworks technology: A DACS state-of-the-art report,” Utica, New York, 1992, https://csiac.org/wp-content/uploads/2021/06/Artificial-Neural-Networks-Technology-SOAR.pdf.

[43] K. Du and M. N. S. Swamy, *Neural Networks and Statistical Learning*. Springer, 2014, https://doi.org/10.1007/978-1-4471-7452-3.

[44] D. W. Marquardt, “An Algorithm for Least-Squares Estimation of Nonlinear Parameters,” *J. Soc. Ind. Appl. Math.*, vol. 11, no. 2, pp. 431–441, 1963, https://doi.org/10.1137/0111030.

[45] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," in *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989-993, Nov. 1994, https://doi.org/10.1109/72.329697.

[46] A. -N. Sharkawy, P. N. Koustoumpardis, and N. Aspragathos, “Human–robot collisions detection for safe human-robot interaction using one multi-input–output neural network,” *Soft Comput.*, vol. 24, no. 9, pp. 6687–6719, 2020, https://doi.org/10.1007/s00500-019-04306-7.

[47] R. Köker, C. Öz, T. Çakar, and H. Ekiz, “A study of neural network based inverse kinematics solution for a three-joint robot,” *Rob. Auton. Syst.*, vol. 49, no. 3–4, pp. 227–234, 2004, https://doi.org/10.1016/j.robot.2004.09.010.

[48] B. Daya, S. Khawandi and P. Chauvet, "Neural network system for inverse kinematics problem in 3 DOF robotics," *2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, pp. 1550-1557, 2010, https://doi.org/10.1109/BICTA.2010.5645269.

[49] E. J.- López, D. S. D. L. M.- Pulido, L. A. R.- Ávila, R. S. D. L. M.- Pulido, J. M.- Campos, and A. A. L.- Martínez, “Modeling of Inverse Kinematic of 3-DoF Robot, Using Unit Quaternions and Artificial Neural Network,” *Robotica*, vol. 39, no. 7, pp. 1230–1250, 2021, https://doi.org/10.1017/S0263574720001071.