CHAOS
Theory and Applications
in Applied Sciences and Engineering

e-ISSN: 2687-4539
RESEARCH ARTICLE
Vol.6 / No.2 / 2024 / pp.90-110
Special Issue (EDIESCA 2023)
https://doi.org/10.51537/chaos.1375866

Investigating Feed-Forward Back-Propagation Neural Network with Different Hyperparameters for Inverse Kinematics of a 2-DoF Robotic Manipulator: A Comparative Study

Rania Bouzid [a,1], **Hassène Gritli** [a,b,2] **and Jyotindra Narayan** [c,3]

[a]Laboratory of Robotics, Informatics and Complex Systems (RISC Lab, LR16ES07), National Engineering School of Tunis, University of Tunis El Manar, BP. 37, Le Belvédère, 1002 Tunis, Tunisia, [b]Higher Institute of Information and Communication Technologies, University of Carthage, Technopole of Borj Cédria, Route de Soliman, BP 123, Hammam Chatt 1164, Ben Arous, Tunisia, [c]Mechatronics and Robotics Laboratory, Department of Mechanical Engineering, Indian Institute of Technology Guwahati, Guwahati 781039, Assam, India.

ABSTRACT Inverse kinematics is a significant challenge in robotic manipulators, and finding practical solutions plays a crucial role in achieving precise control. This paper presents a study on solving inverse kinematics problems using the Feed-Forward Back-Propagation Neural Network (FFBP-NN) and examines its performance with different hyperparameters. By utilizing the FFBP-NN, our primary objective is to ascertain the joint angles required to attain precise Cartesian coordinates for the end-effector of the manipulator. To accomplish this, we first formed three input-output datasets (a fixed-step-size dataset, a random-step-size dataset, and a sinusoidal-signal-based dataset) of joint positions and their respective Cartesian coordinates using direct geometrical formulations of a two-degree-of-freedom (2-DoF) manipulator. Thereafter, we train the FFBP-NN with the generated datasets using the MATLAB Neural Network Toolbox and investigate its potential by altering the hyperparameters (e.g., number of hidden neurons, number of hidden layers, and training optimizer). Three different training optimizers are considered, namely the Levenberg-Marquardt (LM) algorithm, the Bayesian Regularization (BR) algorithm, and the Scaled Conjugate Gradient (SCG) algorithm. The Mean Squared Error is used as the main performance metric to evaluate the training accuracy of the FFBP-NN. The comparative outcomes offer valuable insights into the capabilities of various network architectures in addressing inverse kinematics challenges. Therefore, this study explores the application of the FFBP-NNs in tackling the inverse kinematics, and facilitating the choice of the most appropriate network design by achieving a portfolio of various experimental results by considering and varying different hyperparameters of the FFBP-NN.

KEYWORDS
Robotic manipulator
Inverse kinematics
Feed-Forward back propagation
Artifical neural network
Hyperparameters
Levenberg-Marquardt algorithm
Bayesian regularization algorithm
Scaled conjugate gradient algorithm
Different datasets
Mean squared error
R-value

INTRODUCTION

Robot kinematics plays a fundamental role in respective robotic research and applications. The progress in inverse kinematics algorithms is of utmost importance for advancing this field, as noted in various studies (Gao *et al.* 2017; Liu *et al.* 2017; Rea Minango and Ferreira 2017). However, conventional approaches for solving inverse kinematics problems frequently face issues with convergence and entail intricate iterative procedures, which can negatively impact the overall efficiency and quality of these algorithms, as highlighted in the work of Reiter *et al.* (2018). Additionally, it has been noted in the research work of Zhao *et al.* (2018) and that of Di Pietro *et al.* (2012) that conventional approaches for solving inverse problems lack a unified equation for describing motion.

Manuscript received: 15 October 2023,
Revised: 30 November 2023,
Accepted: 5 December 2023.

[1]rania.bouzid@istic.ucar.tn
[2]grhass@yahoo.fr (**Corresponding author**)
[3]n.jyotindra@gmail.com

Compared to forward kinematics equations, inverse kinematics equations present more significant challenges. Solving inverse kinematics difficulties is typically more complex than solving forward kinematics ones Bouzid *et al.* (2024d,c). The purpose of forward kinematics is to identify the end-effector's position based on joint angles or positions (Bouzid *et al.* 2023, 2024b). In most cases, this can be accomplished using simple geometric calculations. However, inverse kinematics entails determining the joint angles or positions that match to a desired end-effector position, which requires solving a set of nonlinear equations. This can be computationally intensive, with complex relationships and limits (Bouzid *et al.* 2024d,c).

Many researchers and organizations have conducted thorough research into inverse kinematics algorithms within the field of robotics (Benavente-Peces *et al.* 2014; Narayan *et al.* 2022; Becerra and Kremer 2011; Narayan and Singla 2017a). These algorithms primarily concentrate on four key aspects: geometric algorithms, analytical algorithms, geometric-analytic algorithms, and numerical algorithms. An analytical algorithm for solving kinematics' inverse problems in palletization manipulator robotics was proposed by Xu *et al.* (2017). While this analytical algorithm offers certain practicality to some possible extent, it was found to encounter the probem of yielding multiple analytical solutions for a single pose, thereby complicating the determination of a unique solution. Practical experience has revealed that while applying the algorithm to robotics offers flexibility, the design process tends to become overly convoluted. This often leads to the need for numerous iterations, resulting in inefficiencies (Abbas *et al.* 2019; Narayan *et al.* 2018).

Therefore, to address the problem of computationally expensive analytical and numerical solutions, researchers have started exploring intelligent solutions in the last few years (Li and Savkin 2018; Mahajan *et al.* 2017). The authors in (Duka 2014) generated training data for the Neural Network by randomly choosing joint angle values and then determining the resulting end-effector position, following circular trajectories through forward kinematics. Furthermore, the paper introduces a method to rescale the input and output data to fall within the $[-1,\ 1]$ range, thereby improving the network's performance. In another research by Dash *et al.* (2017), the Levenberg-Marquardt (LM) algorithm was used to train the designed network over a set number of epochs. This study focuses on tackling the problem of solving the inverse kinematics of a 6-DoF system through the application of an artificial neural network (ANN). In the work by Mahajan *et al.* (2017), a neural network model was presented, capable of independently governing the actions of a manipulator, thus obviating the need for external guidance. The neural network was trained through unsupervised learning methods, focusing on a 2-DoF system. The primary focus of the study is to trace a circular path and intercept a moving ball. In the study by Narayan and Singla (2017a), the researchers used the adaptive neuro-fuzzy inference system (ANFIS) with a Gaussian membership function to solve the inverse kinematics problem of a 4-DoF SCARA robot, combining fuzzy inference systems and neural network approaches from prior work.

Li and Savkin (2018) proposed a solution using competitive neural networks to address the inverse kinematics problem in robotics, focusing on the task of a mechanical arm grabbing an object. A MATLAB-based simulation was carried out, as referenced in (Kumar *et al.* 2018), to assess the efficacy of an intelligent technique in rectifying position errors during the execution of a circular trajectory by a 2R robot. By introducing minor variations in link lengths, geometric discrepancies were examined by generating a simulated dataset using the kinematic models derived. Subsequently, the neural network was trained on this dataset to forecast position error values within the operational area of the robot. In (Lathifah *et al.* 2018), authors explored an ANN to solve the problem of an inverse kinematics for a 3-link planar serial robotic manipulator. The trained neural network was tested by considering that the robotic manipulator performs square and/or triangle motions within the admissible working-space (Bouzid *et al.* 2024d). Moreover, the LM algorithm trained the neural network for the inverse kinematics problem's solution. Various network architectures were tested by Handayani *et al.* (2018) to find the optimal solution. The proposed method was evaluated using a simple planar manipulator performing tasks such as drawing a square and a triangle, and the results demonstrate the validity of the neural network trained with the Bayesian Regularization (BR) algorithm for solving the inverse kinematics problem. Theofanidis *et al.* (2018) have introduced a novel neural network-based approach for estimating a kinematically redundant robotic arm's forward kinematics and testing it for different configurations.

The neural network is trained in the work by Dumitriu *et al.* (2020), using the LM algorithm to map the manipulator's joints and the end-effector's position based on forward kinematics calculations. The network is trained for different scenarios, and the Mean Square Error (MSE) is used to evaluate the accuracy of the results. The research work in (Gao 2020) demonstrated that the proposed algorithm of the inverse kinematics, which employs an enhanced BPNN, outperforms better than traditional/classical algorithms for inverse solutions when handling with the inverse kinematics problem in manipulator robots with six degrees of freedom (Bouzid *et al.* 2024d). Aravinddhakshan *et al.* (2021) introduced a neural network-based approach for a 5-DoF manipulator through supervised learning, achieving accurate inverse kinematics and optimizing path planning during pick and place operations. This highlights the effectiveness of neural networks in manipulator control. Furthermore, the research of Köker *et al.* (2004) focused on a three-joint robotic manipulator and utilized simulation software to plan cubic trajectories and define the manipulator's work volume. A key strength highlighted in the study is the neural network's remarkable online performance. In addition, multiple neural networks were employed in (Takatani *et al.* 2019) to learn the inverse kinematics of redundant robotic manipulators using an independent approach, and by studying different structures of the evaluation function (Bouzid *et al.* 2024d). Furthermore, the training data employed in this learning methodology consists of different endpoints, different postures, and different evaluation values of the robotic manipulators. Meanwhile, Ibarra-Pérez *et al.* (2022) emphasizes the challenge of setting structural parameters for neural networks, advocating for optimization-based methods over trial-and-error, saving time and improving performance. Lastly, in the work of Aysal *et al.* (2023), machine learning techniques are found to be a viable option for analyzing the kinematics of a 3-DoF robot arm with an RRR design, mainly using an MLP model to ensure system stability.

In (Wagaa *et al.* 2023), various Deep Learning networks were developed to solve the inverse kinematics of 6-DoF manipulator robots. ANN, Convolutional Neural Network (CNN), Long-Short Term Memory (LSTM), and Gated Recurrent Unit (GRU) are the neural network architectures that have been considered. Furthermore, authors examined the performance of analytical and neural systems in producing robot trajectories using the RoboDK simulator to display simulation results with real-world implications. In (Cagigas-Muñiz 2023), various strategies involving ANNs were

proposed and studied. The results revealed that the proposed original bootstrap sampling and hybrid methods could significantly outperform approaches that just use one ANN. Nevertheless, none of these advancements completely solved the inverse kinematics problem in articulated robots. García-Samartín and Barrientos (2023) addressed the forward kinematics' problem using ANNs and Genetic Algorithms (GA). Using the publicly available Inverse Kinematic (IK) model, both GA and ANN approaches were implemented. Authors showed, compared to another approach, that the proposed methodologies produced equivalent or higher results in terms of accuracy and time. Authors in (Bouzid *et al.* 2023, 2024b) studied the forward kinematics problem of a 2-DoF robotic manipulator via ANNs. Moreover, they tackled in (Bouzid *et al.* 2024a) the same problem for a SCARA manipulator robot. Furthermore, the problem of studying and solving the issues of the inverse kinematics for the manipulator robot was considered in (Bouzid *et al.* 2024d,c). Other solving approaches of the inverse kinematics problem for articulated manipulator robots can be found within these previous references.

The existing literature on the subject reveals a noticeable gap in research related to the comparative analysis of neural networks with different hyperparameters when applied to solving the inverse kinematics of robot manipulators. As a response to this deficiency, our study makes a substantial contribution by introducing a Feed-Forward Back-Propagation Neural Network (FFBP-NN) specifically designed to tackle the inverse kinematics problem of a 2-DoF articulated robotic manipulator. Through the process, we investigated the effectiveness of the FFBP network architecture with three distinct generated datasets: a random-step-size dataset, a fixed-step-size dataset, and a sinusoidal-signal-based dataset with varying frequencies (Bouzid *et al.* 2024d,c, 2023, 2024b). Moreover, for each dataset type, the network architecture is tested with different hyperparameters, such as the number of hidden layers, neurons in the hidden layer, and three different training optimization algorithms, namely the Levenberg-Marquardt (LM) (Ranganathan 2004), the Bayesian regularization (BR) (Kayri 2016), and the Scaled conjugate gradient (SCG) (Møller 1993). The testing and verification phases are achieved in order to evaluate the capacity of the trained neural network to minimize approximation errors and appropriately estimate inverse kinematics.

The rest of this paper is structured like so: Section 2 delves into the mathematical representation of the forward kinematics of manipulator robots with n degrees of freedom, while Section 3 explores the computational aspects of the inverse kinematics for a 2-DoF manipulator robot. Section 4 introduces a novel approach using an FFBP-NN to efficiently solve inverse kinematics problems, detailing the network's architecture. In this previous Section, we provide a brief overview of FFBP-NNs to ensure reader understanding. A flowchart and the hyperparameters used in the proposed FFBP-NNs are also illustrated in this previous same section. Section 5 presents numerical results obtained through the FFBP-NNs' application and initiates a discussion on implications, potential improvements, and broader applications within the field of robotics. The paper concludes with a conclusion and some possible future directions for improvements, presented in Section 6.

FORWARD KINEMATICS OF n-DOF MANIPULATOR ROBOTS

It is worth mentioning first that the terminology DoF stands for "Degree of Freedom". In the context of robotics, mechanics, and other related disciplines, it refers to the number of independent parameters or variables that define the configuration or motion of a system. Thus, in the context of a robot arm, the number of degrees of freedom would represent the number of independent ways the arm can move or rotate.

A Brief Description on Forward Kinematics of n-DoF Manipulator Robots

Forward kinematics of n-DoF manipulator robots is a fundamental concept in robotics that deals with determining the position and orientation of the robot's end-effector (usually a tool or gripper) in the workspace based on the joint angles or variables of the robot's individual links. It is akin to tracing the path of a robot's "hand" as it moves through its various joint configurations. Methods from geometry and linear algebra, trigonometric transformation matrices, and homogeneous coordinate transformations are frequently used in traditional solutions to the forward kinematics problem. Here is a brief overview of the conventional solution approaches for the computation of the forward kinematics of manipulator robots:

1. *Geometric Approach*: It offers a straightforward understanding of how a robot's joints and links affect its end-effector's position and orientation, useful for simpler robots like planar ones but less effective for complex structures with closed-loop chains or redundancy due to accuracy challenges and lack of closed-form solutions (Kim *et al.* 2016).

2. *Trigonometric methods*: While excelling in simplicity and computational efficiency, trigonometric methods are most suitable for planar robots because they provide analytical solutions without iterative techniques (Petrescu *et al.* 2017). However, they may not be as effective for complex robots in three-dimensional spaces or with unconventional joint arrangements, as their assumptions may lead to reduced accuracy. Engineers and roboticists should assess their suitability for specific applications and consider alternative approaches when dealing with intricate systems or non-standard geometries.

3. *Coordinate transformations*: Coordinate transformations have versatile applications in various robotic systems, including 2D and 3D environments with different degrees of freedom (Wang *et al.* 2014). They provide a mathematically rigorous foundation, enhancing complex robots' capabilities and integrating seamlessly with other techniques. However, implementing them can be intricate, especially for robots with many joints and complex link geometries, potentially leading to longer development times and errors. Proficiency in coordinate transformations may require a robust mathematical background, posing a learning curve for some robotic practitioners.

Extending the concept of coordinate transformation, forward kinematics is calculated using Denavit-Hartenberg (DH) parameters and the homogeneous transformation matrix (Denavit and Hartenberg 1955). The DH parameters provide a systematic way to describe the geometric relationship between the robot's successive joints and links (Narayan and Singla 2017a). The Homogeneous Transformation Matrix combines DH parameters to express the transformation from one coordinate frame (associated with a specific joint) to another, effectively mapping the position and orientation of each link concerning the previous one. By multiplying these transformation matrices sequentially from the base link to the end-effector, the forward kinematics algorithm computes the final transformation that represents position of the robot's end-effector in the base coordinate frame.

Furthermore, for each joint/frame, the parameters in this previous homogeneous transformation matrix, noted as ${}_{i-1}^{i}\mathcal{H}$, are defined, according to the DH method, like so (Ganapathy 1984):

- Link length (a): The path of the shared normal defining the difference between the preceding z-axis (that is of the $(i-1)$th frame) and the actual z-axis (that is of the ith frame).
- Link twist (α): The angle about the shared normal between the preceding z-axis and the present z-axis.
- Link offset (d): The path from the previous x-axis to the actual x-axis, along the preceding z-axis.
- Joint angle (θ): The degree of rotational angle about the z-axis between the preceding x-axis and the actual x-axis.

The homogeneous transformation matrix ${}_{i-1}^{i}\mathcal{H}$ from the $(i-1)$th frame to the next one (ith frame) is represented via the following expression (Ganapathy 1984):

$$ {}_{i-1}^{i}\mathcal{H} = \begin{bmatrix} {}_{i-1}^{i}\mathcal{R} & {}_{i-1}^{i}\mathcal{T} \\ \mathcal{O} & 1 \end{bmatrix} \tag{1} $$

where ${}_{i-1}^{i}\mathcal{R}$ and ${}_{i-1}^{i}\mathcal{T}$ are, respectively, the rotation and translation matrices from one frame to the next one. Moreover, the symbol $\mathcal{O}$ stands for the zero matrix with appropriate dimension.

Such homogeneous transformation matrix (1) is explicitly expressed in terms of the DH parameters, θ, d, α and a, as follows:

$$ {}_{i-1}^{i}\mathcal{H} = \begin{bmatrix} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & a\cos(\theta) \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

The Adopted 2-DoF Robotic Manipulator and its Forward Kinematics

In our study, we utilize a 2-DoF manipulator robotic model as shown in Figure 1. In the kinematic description of the robotic system under consideration, the joints are denoted as joint 1 and joint 2, representing the rotational angles θ_1 and θ_2, respectively. The associated links are labeled as follows: link 0 corresponds to l_0, link 1 to l_1, and link 2 to l_2. This notation establishes a clear and concise representation of the joint angles and link lengths, facilitating the systematic analysis of the robot's kinematics.

By following the steps and the implementation of the geometrical method and validating with the DH method, the forward kinematics equations for the 2-DoF manipulator robot can be obtained as follows (Ghaleb and Aly 2018):

$$ X = l_0 + l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2) \tag{3} $$
$$ Y = l_1\sin(\theta_1) + l_2\sin(\theta_1+\theta_2) \tag{4} $$

For our specific scenario, the lengths of the 2-DoF manipulator robot are set as follows: $l_0 = 1\ [m]$, $l_1 = 2\ [m]$, and $l_2 = 3\ [m]$. Note that in these equations (3) and (4), the parameters θ_1 and θ_2 represent the joint angles of the 2-DoF manipulator robot, serving as input parameters for computing the Cartesian coordinates X and Y.

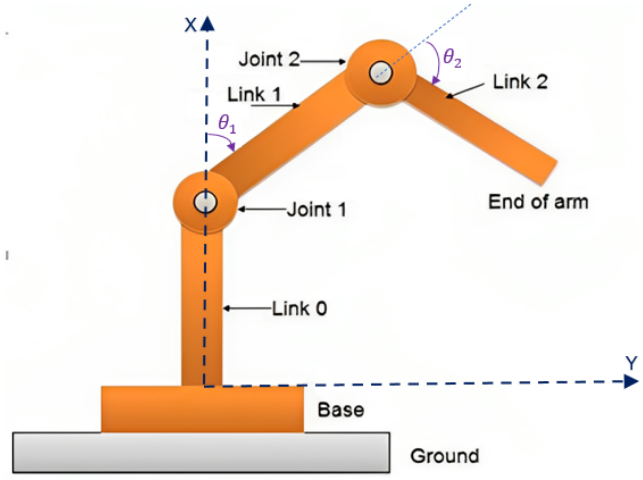


Figure 1 Schematic of a 2-DoF manipulator robot (updated from (Madhuraghava *et al.* 2018)).

INVERSE KINEMATICS OF THE 2-DOF MANIPULATOR ROBOT

In the case of inverse kinematics for the 2-DoF manipulator robot, the problem is formulated to calculate the joint angles necessary to position the end-effector at a specific Cartesian coordinate in the workspace, as shown in Figure 2. Thereafter, an intelligent neural network-based solution is proposed to address the problem formulation.

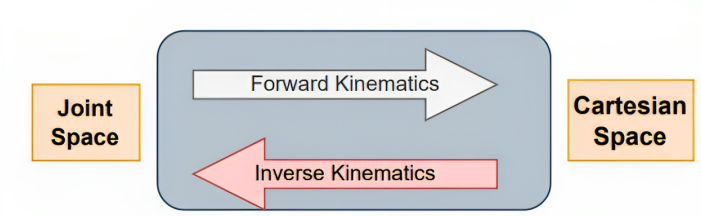


Figure 2 Schema of the Forward/Inverse Kinematics. The left-hand side of the figure represents the manipulator articulation variable. The right-hand side represents the location of the robot end-part.

Problem Formulation

The complexity arises from the nonlinear and intricate mathematical equations governing the connection between joint angles q and the resulting end-effector position Z, modeling the following forward kinematics model (Narayan and Singla 2017b):

$$ Z = \mathcal{F}(q), \qquad q \in \mathbb{R}^n, \qquad Z \in \mathbb{R}^m \tag{5} $$

and therefore the following inverse kinematics model, which is considered to be unknown:

$$ q = \varphi(Z) \tag{6} $$

Here are a few problems associated with the inverse kinematics:

1. *Nonlinearity*: Nonlinear inverse kinematics challenges arise due to complex joint configurations or irregular manipulator robot shapes, necessitating computationally intensive solutions through iterative numerical methods or optimization techniques (Snieder 1998).

2. *High-dimensional Spaces*: As the number of joints in a manipulator robot increases, the dimensionality of the inverse kinematics problem also increases. To tackle this problem, dimension reduction and advanced optimization methods may be required (Petrović 2018).

3. *Joint limits*: Manipulator robots typically have joint limits that restrict the range of motion for each joint. Ensuring that the solutions of the inverse kinematics respect these physical limits is essential to prevent damage to the manipulator robot (Huo and Baron 2008).

Proposed Method

In this research work and to solve the inverse kinematics of the 2-DoF manipulator robot, the proposed approach is articulated around the following points:

- Our research project centers on applying intelligent techniques to tackle inverse kinematics challenges, particularly on ANNs.
- In our investigation, we will explore the use of FFBP-NNs utilizing various training optimizers and diverse numbers of hidden layers.
- We aim to evaluate their effectiveness across three different datasets to enhance our understanding of their capabilities.

FEED-FORWARD BACK-PROPAGATION NEURAL NETWORK FOR SOLVING THE INVERSE KINEMATICS PROBLEM

A Brief Overview on Feed-Forward Back-Propagation Neural Networks

An FFBP-NN stands as a pivotal construct within the expansive realm of ANNs, designed primarily to excel in supervised machine learning endeavors such as classification and regression, prediction of time series, as well as modeling of complex nonlinear dynamical systems (Cimen *et al.* 2019; Martinez-garcia *et al.* 2022; Karaca 2023; Keleş *et al.* 2023; Noorani and Mehrdoust 2022). An FFBP-NN fundamentally comprises a layered architecture where neurons are intricately linked, guiding information in a carefully designed unidirectional path. This complex network structure typically includes an input layer, which receives initial data inputs; a series of hidden layers, one or more in the count, actively involved in intermediate data processing and the extraction of vital features; and, to complete the sequence, an output layer tasked with generating the network's predictions. The core of this neural architecture hinges on two essential components: the allocation of weights to neural connections, which indicates the connection strength, and the inclusion of bias terms at each neuron, discreetly introducing an adjustment to the neuron's activation function output.

The crux of FFBP-NN operation is the feedforward process, wherein data systematically traverses the network's layers. Within each layer, a neuron undertakes the intricate task of computing a weighted sum of its inputs, adding the bias term into the equation, and applying a designated activation function, resulting in an output that cascades to the next layer. These activation functions, spanning the spectrum from sigmoid and hyperbolic tangent to the robust rectified linear unit (ReLU), serve to imbue the network with a capacity for nonlinearity, thereby empowering it to model intricate and non-trivial data relationships effectively. We find the eagerly awaited predictions at the apex of the network, located within the output layer. An essential aspect of this procedure entails the utilization of a loss or cost function, which scrupulously quantifies the gap between these predictions and the pristine target values present in the training dataset. The optimization process, a crucial stage in an FFBP-NN, is guided by the venerable back-propagation algorithm.

This iterative mechanism recalibrates the network's core parameters - the weights and biases - in pursuit of the singular objective: minimizing the loss function. Underpinning this algorithmic operation is the gradient descent technique, a stalwart of optimization methodologies, masterfully guiding parameter adjustments. Central to this process is the learning rate, a hyperparameter wielding influence over the scale of parameter updates, thereby determining the network's convergence rate. With the profound complexity and nuance embedded within FFBP-NNs, hyperparameters serve as vital navigational coordinates in this intricate journey. These encompass many facets, encompassing the count of hidden layers, the number of neurons nested within each layer, the judicious selection of activation functions, and the meticulous tuning of the learning rate. Through these careful adjustments, FFBP-NNs evolve from their nascent state into formidable models, attaining an esteemed position within the pantheon of machine learning paradigms. FFBP-NNs play a pivotal role across multifarious domains, underscoring their enduring relevance and adaptability through their ability to capture, decode, and illuminate intricate data relationships.

Training With Three Different Datasets

The proposed method employs three distinct datasets: (1) a dataset with a fixed step size, (2) another with random step sizes, and (3) a sinusoidal-signal-based dataset featuring varying frequencies. The selection of these datasets aims to thoroughly evaluate the FFBP-NN's performance across various input data types.

It is important to note that we arbitrarily use the parameters, and multiple possible combinations depend on our choices. We have discovered these results through numerical experimentation, but obtaining other values and even more efficient models is possible. We conducted tests over 1000 epochs, exploring using 1 to 5 hidden layers for a given number of layers (as shown in Figure 3), and a specific dataset size.

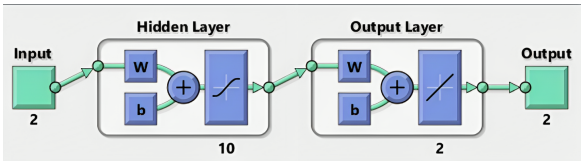
Fixed-step-size Dataset: This initial dataset served as the main foundation for training ANNs through a FFBP method. This dataset was meticulously curated to evaluate the ANN's performance across various scenarios (Bouzid *et al.* 2023, 2024d,b,c). The dataset creation process involved systematically altering the values of θ_1 and θ_2, incrementing them by a fixed step size h equal to $h = 0.02$, within the interval from $\theta_i^{\min} = -\pi$ to $\theta_i^{\max} = \pi$, for $i = 1, 2$.

Thus, for each articulated variable θ_i, we build the vector of all values of θ_i by sweeping the interval $[\theta_i^{\min} : \theta_i^{\max}]$, for all $i = 1, 2$, for left to right with the fixed step $h = 0.02$. The following expressions are used to build such intervals of the two parameters θ_1 and θ_2.

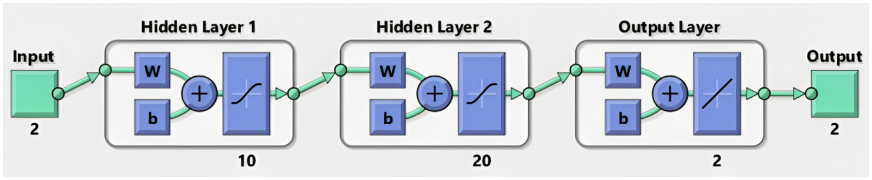
$$\theta_1^{\text{Range}} = [\theta_1^{\min} : h : \theta_1^{\max}] \tag{7}$$

$$\theta_2^{\text{Range}} = [\theta_2^{\min} : h : \theta_2^{\max}] \tag{8}$$

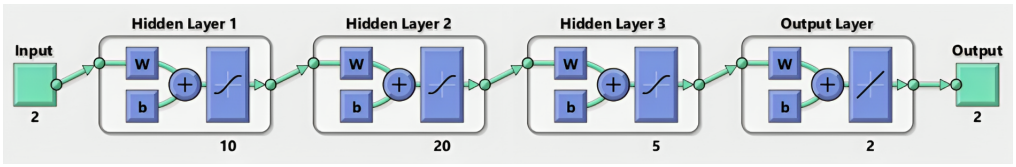
The size of each vector θ_i^{Range}, with $i = 1, 2$, is equal to $N = \frac{2\pi}{h} + 1 = 315$. It is worth to note that the decrease of the value of the parameter h will lead to the increase of the value of N, and hence of the size of the dataset in question, which is equal in the present case (that is with $h = 0.02$) to $N^2 = 99225$.



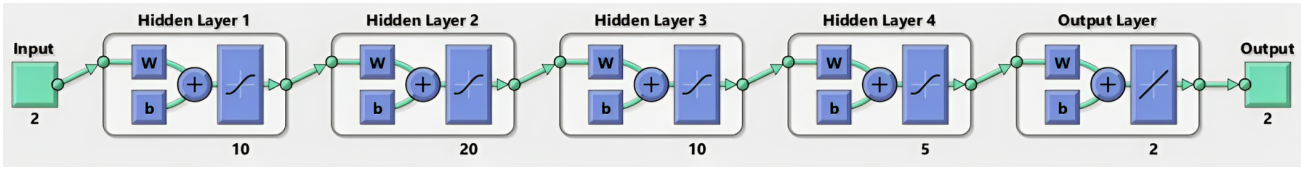
(a)



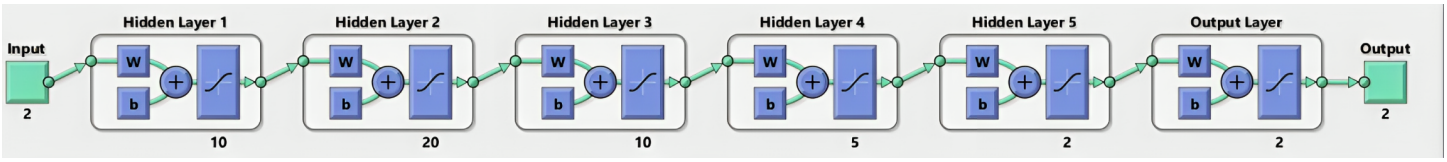
(b)



(c)



(d)



(e)

Figure 3 Different adopted ANN architectures with a different number of neurons in each hidden layer: (a) ANN model with only one hidden layer having 10 neurons, it will be noted as 1 Hidden layer (10), (b) ANN model with two hidden layers where the first hidden layer has 10 neurons, whereas the second hidden layer has only 2 neurons, it will be referred to as 2 Hidden layers (10,2), (c) ANN model with three hidden layers, where the number of neurons in each hidden layer is successively 10, 20 and 2, it will be noted as 3 Hidden layers (10,20,2), (d) ANN model with four hidden layers and where the number of neurons associated to each layer is respectively 10, 20, 5, and 2, and it will be referred to as 4 Hidden layers (10,20,5,2), and (e) ANN model with five hidden layers, and it will be indicated as 5 Hidden layers (10,20,10,5,2).

For each possible combination of these joint angles θ_i in the interval θ_i^{Range}, for all $i = 1, 2$, by sweeping then from left to right, we employed the forward kinematics equations (3) and (4) to compute the corresponding Cartesian coordinates (X and Y). Therefore, the dataset consisted of pairs of input and output data, where the Cartesian coordinates represent the outputs, whereas the inputs are the two joint angles (θ_1 and θ_2).

The main goal is to enable the FFBP-NN to accurately predict joint angles from various end-effector positions. We achieved this by using diverse input and output data in the dataset and the FFBP-NN during training to improve the proposed ANN's ability to generalize and make precise predictions of the positions of the adopted 2-DoF manipulator robot.

Random-step-size Dataset: This dataset was deliberately designed to inject an element of unpredictability and variability into the input data, serving as a test of the FFBP-NN's adaptability and resilience. The primary objective behind creating this dataset was to gauge the FFBP-NN's capacity to handle unforeseen patterns and assess its robustness when confronted with such unanticipated scenarios (Bouzid *et al.* 2023).

To generate the random dataset for the inverse kinematics of the 2-DoF manipulator robot, we harnessed the power of the "rand" function to produce random values for the joint angles, θ_1 and θ_2. Each data point within this dataset was characterized by two joint angles, θ_1 and θ_2, randomly chosen from a non-uniform distribution (Bouzid *et al.* 2023, 2024b). We applied a scaling and shifting technique to ensure that these randomly generated joint angles fell within the desired range of $-\pi$ to π. Specifically, we multiplied the random values by 2π and subtracted π, yielding joint angles that span the entire range from $-\pi$ to π (Bouzid *et al.* 2023, 2024d,b).

The following expression elucidates the computation process

for this random dataset (Bouzid *et al.* 2023, 2024d,b):

$$\theta^{\text{Range}} = \text{rand}(N, 2) \times 2\pi - \pi \tag{9}$$

where N is the desired total number of parameters, which gives hence the size of the dataset. Note that this expression (9) will generate the complete dataset for the two variables θ_1 and θ_2. The output of this equation is then a vector composed of two columns: the first column is for θ_1, whereas the second column is for θ_2.

Then, once the random part/subset of the dataset is generated, we compute the two Cartesian coordinates X and Y of the manipulator robot using the forward kinematics equations (3) and (4). As a result, we obtain a matrix with two columns where the first column is for the X coordinate and the second column is for the Y coordinate. This matrix is therefore saved in the dataset along with the first part composed of the two variables θ_1 and θ_2. Thus, the resulting matrix is composed of four columns and N rows. For this random dataset, we fixed $N = 1000$.

The inherent randomness and variability of this adopted dataset pose a robust challenge to the FFBP-NN, forcing it to adapt and learn from unforeseen patterns. Using previously generated joint angles, we systematically cycled through forward kinematics equations for each data point. This iterative process created a dataset with diverse and random combinations of Cartesian coordinates and joint angles. Each entry in the dataset displayed a unique combination of joint angles associated with Cartesian coordinates, enabling a comprehensive and random exploration of the end-effector positions of the manipulator robot (Bouzid *et al.* 2023, 2024d,b,c).

Sinusoidal-signal-based Dataset: We harnessed the capabilities of the FFBP-NN by introducing a third dataset tailored to evaluate its proficiency in handling sinusoidal signals characterized by varying frequencies. This dataset was pivotal in assessing the performance of the FFBP-NN when confronted with diverse frequency patterns. Our primary goal was to comprehensively examine how effectively the FFBP-NN could discern and predict the intricate cyclic patterns inherent in the input data. This exploration was instrumental in enabling the FFBP-NN to acquire an in-depth understanding of these complex signals and enhance its ability to generalize across a broad spectrum of cyclic patterns (Bouzid *et al.* 2023, 2024d,b,c).

The sinusoidal-signal-based dataset encompassed two sinusoidal signals with distinct characteristics, as outlined in Table 1. Each signal i corresponds to the joint variable θ_i, for $i = 1, 2$.

This dataset facilitates the creation of visualizations for tracking the fluctuations in joint angles θ_1 and θ_2 concerning an angular parameter. These visualizations are constructed using the following equations:

$$\theta_1 = A_1 \times \sin(f_1 \times \xi_1 + \phi_1) \tag{10}$$
$$\theta_2 = A_2 \times \sin(f_2 \times \xi_2 + \phi_2) \tag{11}$$

In these equations (10) and (11), and in order to generate the two sinusoidal signals, the two parameters ξ_1 and ξ_2 are varied within the admissible interval $[-\pi, \pi]$. As a result, another form of the distribution of the values of the two angles θ_1 and θ_2 will be obtained. A such distribution is entirely different to that of the first and second datasets.

When we generate the parameters and equations for the sinusoidal-signal-based dataset, a figure emerges as its representation. This figure encompasses two subplots illustrating the generated curves of our variables, which are θ_1 and θ_2 (see Figure 4). In the initial subplot at the top, the curve is depicted in blue, illustrating the connection between the angle on the x-axis (ξ) and the corresponding joint angle (θ_1) on the y-axis. This subplot provides insights into how the joint angle θ_1 varies with respect to the angle/variable ξ. In the second subplot at the bottom, the curve is presented in magenta. Here, the x-axis (ξ) signifies the angle, while the y-axis represents the joint angle (θ_2). This subplot enables us to examine how the joint angle θ_2 responds to changes in the angle/variable ξ. Collectively, these subplots offer a visual depiction of the sinusoidal-signal-based dataset, highlighting the interplay between angles and their corresponding joint angles, as demonstrated in Figure 4. Additionally, this visual representation aids in understanding the dataset's characteristics.

Table 1 Parameters used for the sinusoidal-signal-based dataset for generating the values of the two angular positions θ_1 and θ_2 of the 2-DoF manipulator robot.

Parameter	Signal 1 of θ_1	Signal 2 of θ_2
Frequency f [Hz]	1.5	10
Phase ϕ [rad]	0	$\pi/4$
Amplitude A [rad]	π	π
Angle ξ [rad]	$[-\pi, \pi]$	$[-\pi, \pi]$
Number of samples N	1000	1000

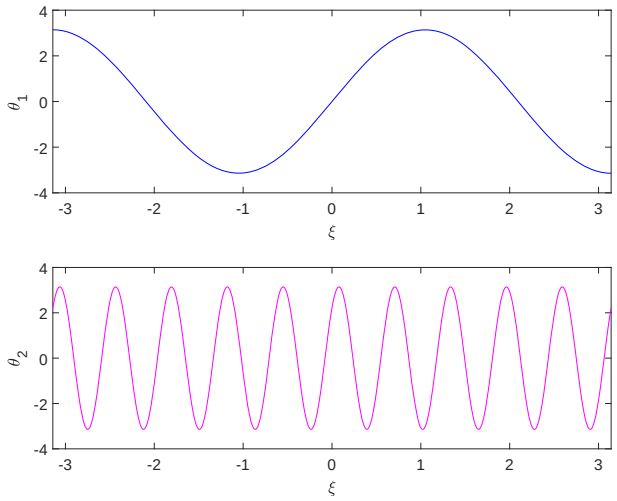


Figure 4 Presentation of the generated signals of the two sinusoidal signals of the two variables θ_1 and θ_2 of the 2-DoF manipulator robot.

Therefore, once the values of two joint angles θ_1 and θ_2 are obtained according to the adopted sinusoidal distributions, we introduce the forward kinematic model, that is equations (3) and (4), for the computation of the Cartesian coordinates X and Y of the 2-DoF manipulator robot. These results of X and Y are then putted together with the generated variables θ_1 and θ_2 to form hence the dataset in question. Such dataset has a size equal to $N = 1000$.

Adopted Training Optimizers for the FFBP-NNs

In the complex world of training neural networks, there are many optimization methods to tweak the inner workings of a model to get better at reducing errors. Within this realm, the tapestry of algorithms is rich, each weaving its own unique pattern in the grand design of optimization. Our work focuses on three of these distinguished algorithms, each bearing its own distinctive character and prowess.

Levenberg-Marquardt (LM) Optimizer: First, the Levenberg-Marquardt (LM) algorithm emerges as a stalwart choice (Ranganathan 2004). Emerging from the challenges of nonlinear least squares problems, this method utilizes a repetitive process of tweaking parameters. It skillfully estimates the Hessian matrix, which plays a crucial role in governing the shape of the error surface, and combines it with a stabilizing factor. This approach excels when dealing with relatively modest-sized networks, achieving rapid and captivating convergences.

Bayesian Regularization (BR) Optimizer: Delving deeper into the optimization domain, the Bayesian Regularization (BR) algorithm emerges as a prudent approach where the role of a Bayes theorem is crucial, steering the training process towards regularization (Kayri 2016). Endowed with prior knowledge concerning the model's parameters, it safeguards against the perilous pitfalls of overfitting. Pioneering a trail where prior probability distributions are important, the algorithm's objective is to unearth the posterior distribution that, with a predestined sense, maximizes the likelihood of the available data. This algorithm offers unwavering performance against noisy or scanty data.

Scaled Conjugate Gradient (SCG) Optimizer: The Scaled Conjugate Gradient (SCG) algorithm takes the stage as an efficient approach where it marries the concepts of conjugate gradients with adaptive step sizes (Møller 1993). The algorithm dynamically scales its step size according to the undulating contours of the error surface. In doing so, it navigates the rugged terrain of optimization with unparalleled grace, achieving swifter convergence. The algorithm's reputation precedes it, proving its mettle in the daunting task of handling large-scale networks. Its legacy lies in computational efficiency and an unwavering commitment to the cause of convergence.

Evaluation Metrics

MSE: In the context of our research, the MSE is used as a the main metric for assessing the accuracy of predictions of the adopted FFBP-NNs. Such metric is calculated using the following expression:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{12}$$

where n is the number of observations, y_i represents the actual values, and $\hat{y}_i$ represents the predicted values. This equation quantifies the average squared difference between the actual and predicted values. The MSE is a valuable tool in evaluating the performance of predictive models. Such metric will be essentially used to compared between the performance of the different adopted artificial neural networks.

Coefficient of correlation R-value: Moreover, to evaluate the performance of the neural network models, we will use the regression metric, the coefficient of correlation R, called also as the coefficient of determination R^2. Such metric R is expressed and computed according to the following relation (del Rosario Martinez-Blanco *et al.* 2016):

$$\mathrm{R} = \frac{\sqrt{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}}{\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{13}$$

where $\bar{y}$ stands for the mean of the actual values.

It is important to note that it is possible to use other metrics to measure the performance of the FFBP-NN models. Such metrics can be the Mean Absolute Error (MAE), the Root Mean Square Error (RMSE), the Mean Absolute Percentage Error (MAPE), the Relative Root Mean Squared Error (rRMSE), the Normalized Mean Square Error (NMSE), the Relative Mean Absolute Error (rMAE), the Mean Biased Error (MBE), the Mean Relative Error (MRE), just to mention a few (del Rosario Martinez-Blanco *et al.* 2016; Darba *et al.* 2022; Keleş *et al.* 2023). In this present work, only the MSE and the R-value are considered as two metrics to evaluate the performance of the FFBP-NN models in solving the inverse kinematics of the 2-DoF manipulator robot.

Solving Methodology of the FFBP-NNs Training Process

In pursuit of transparency and reproducibility, we adopted MATLAB as our primary software platform, harnessing a suite of specialized toolboxes to streamline implementation and experimentation. The Neural Network Toolbox (NNT) within MATLAB played a pivotal role in this pursuit, offering crucial functionalities for the design, training, and simulation of neural networks. The NNT in MATLAB is a versatile collection of tools tailored for a spectrum of applications, ranging from pattern recognition to time-series prediction. Noteworthy features include its support for diverse neural network architectures, encompassing feedforward networks, radial basis networks, and self-organizing maps. The toolbox empowers users to define network structures with ease, specifying layers, nodes, and activation functions. MATLAB's NNT further stands out with its array of training algorithms, including Levenberg Marquardt and Bayesian regularization, allowing us to fine-tune parameters for optimal results. Moreover, its graphical tools facilitate the visualization of neural network architectures, enabling us to analyze performance through plots and confusion matrices.

In this work, and in order to solve the inverse kinematics of the 2-DoF manipulator robot using ANNs models, we followed some specific steps. The flowchart of Figure 5 reveals these different and specific steps followed to train the adopted FFBP-NNs using the proposed datasets arriving to the final step, which is the displaying of simulation results by plotting the MSE, the regression curves and the error histograms.

Hyperparameters for the Training Process

Table 2 reveals the different parameters and hyperparameters used for the training of the proposed FFBP-NNs and their values. For each set of (hyper-)parameters, the training process is executed according to the flowchart presented in Figure 5. The adopted FFBP-NN models are illustrated in Figure 3, where we used an architecture composed of 1, or 2, or 3, or 4, or 5 hidden layers. For each hidden layer, it corresponds the number of neurons.

In this work, two kinds of the activation function have been considered in the training phase: (1) the hyperbolic tangent sigmoid transfer function (tansig) used for all hidden layers, and (2) the linear transfer function (purelin) for the output layer (see (Keleş *et al.* 2023) for for further details about these two activation

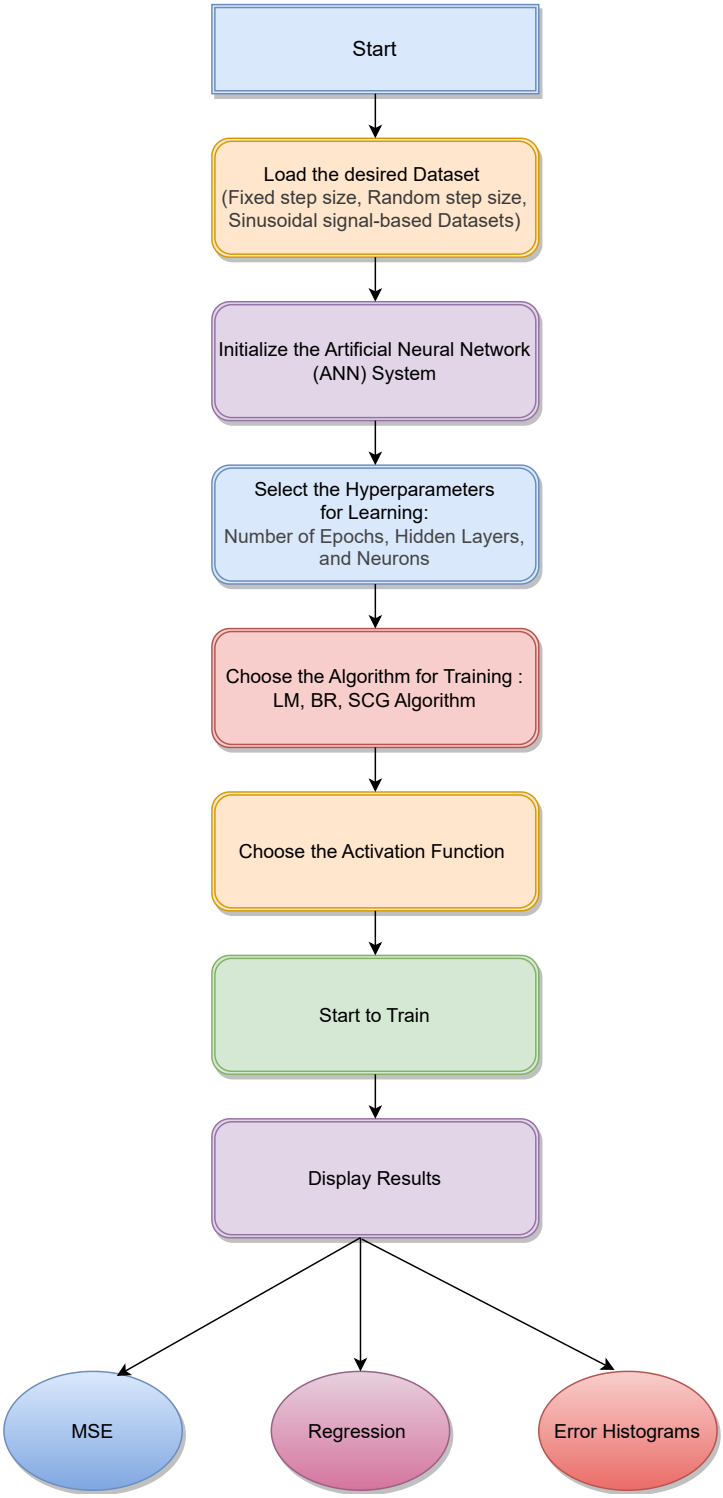


Figure 5 Flowchart describing the procedure for training the FFBP-NN and displaying graphical results.

Table 2 Parameters/hyperparameters, and their adopted values, used for the training process of the proposed FFBP-NNs.

Parameter	Value
Number of inputs	2
Number of outputs	2
Number of hidden layers (NbHLs)	1 or 2 or 3 or 4 or 5
Number of neurons for each layer (NbNs)	NbHLs: 1 — NbNs: 10 NbHLs: 2 — NbNs: 10, 20 NbHLs: 3 — NbNs: 10, 20, 5 NbHLs: 4 — NbNs: 10, 20, 10, 5 NbHLs: 5 — NbNs: 10, 20, 10, 5, 2
Activation function of the hidden layers	tansig
Activation function of the output layer	purelin
Optimizer	LM or BR or SCG
Maximum validation failures	50
Minimum gradient	10^{-6}
Training goal	10^{-6}
Maximum number of epochs	1000

functions). Although it is possible to select the activation function tansig for the output layer; however, no clear enhancement of the results has been observed.

Remark 1. *It is worth to note that in the present work, we have adopted five different architectures of the FFBP-NN, as seen in Figure 3. For each model, we increased the number of the hidden layer, for which the first model has only one hidden layer, whereas the last and fifth one has five hidden layers. However, the number of neurons for each hidden layer is entirely arbitrary. Actually, we tried with other possibilities of the number of neurons, but we have not observed a clear difference in the MSE results. Moreover, the type of the activation functions for the hidden layers and the output layer is fixed for all models. As noted previously, by selecting a tansig activation function for the output layer, we have not noted a clear difference in the simulation results.*

RESULTS AND DISCUSSIONS

In this section, we will show some simulation results of the training of the adopted FFBP-NN architectures using the three optimizers: the Levenberg-Marquardt (LM) optimizer, the Bayesian Regularization (BR) optimizer, and the Scaled Conjugate Gradient (SCG) optimizer. Moreover, the values of the different hyperparameters, along with these optimizers, are given in Table 2. As noted previously, the MSE will be used as the main metric for the evaluation of the performance of the training process and hence of the FFBP-NN models illustrated in Figure 3.

Feed-Forward Back-Propagation Neural Network with the Levenberg-Marquardt (LM) Algorithm

When assessing the training performance of the FFBP-NN using the LM optimization algorithm across a diverse range of datasets – fixed, random, and sinusoidal – it becomes evident that their performance exhibits relatively significant variations. The outcomes of these experiments are visually presented in Figure 6.

In the case of the fixed-step-size dataset, the FFBP-NN model demonstrated its most favorable validation MSE of 2.3255, achieving this at the 116th epoch. This specific neural architecture comprised one hidden layer with 10 neurons. In contrast, when the FFBP-NN model was trained using the random-step-size dataset, its highest validation MSE of 2.2345 was observed at the 75th epoch. The FFBP-NN architecture featured four hidden layers with (10, 20, 5, 2) neurons in this scenario. Lastly, when the FFBP-NN model was trained with a sinusoidal-signal-based dataset, it yielded the best validation MSE of 2.4656 at the 41st epoch. Similarly, this model employed a neural architecture with four hidden layers having (10, 20, 5, 2) neurons, as detailed in the accompanying Table 3.

Comparing these results, it is evident that the FFBP-NN model's performance varies significantly depending on the nature of the dataset used for training. When trained on a dataset with random step-size values, the model demonstrated its best performance regarding the lowest validation MSE. This indicates its ability to generalize and adapt to the irregular patterns within the random dataset. However, when exposed to a dataset based on sinusoidal signals, the performance of the FFBP-NN model declined, as shown by a significantly higher validation MSE. This observation suggests that the model faced difficulties in effectively capturing the complex and oscillatory nature of the sinusoidal data. In contrast, when the model was trained on a dataset with a fixed step size, it exhibited an even higher validation MSE. This indicates that the FFBP-NN model excelled at capturing the inherent patterns within this specific data distribution.

The neural architecture used in these experiments consistently had two hidden layers with varying neuron configurations, except for the sinusoidal dataset, which had four hidden layers. This architectural difference didn't have a straightforward correlation with model performance, as seen in the varying MSE values across the three different datasets. Therefore, the dataset choice significantly impacts the model performance more than the specific neural architecture.

When training an FFBP-NN model in MATLAB for regression, it is common to split the dataset into three subsets: training, validation, and test. The training set is the largest and forms the foundation for model development, involving weight and bias adjustments based on prediction errors and a chosen regression loss function like MSE. The validation set, smaller in size, is used to evaluate the model's performance during training, helping to identify overfitting or underfitting and compute key regression metrics, including MSE. Finally, the test set, a distinct data subset untouched during training or parameter adjustments, rigorously evaluates the model's performance. It tests the model on unseen data, calculating regression metrics to assess its ability to make precise predictions. By dividing data into training, validation, and test sets, an evaluation framework is created to select the best model based on validation performance and provide an unbiased, comprehensive evaluation of the test set for overall efficacy.

The results obtained for regression are graphically represented in Figure 7. The first dataset, using a fixed step size, had regression coefficients of $R = 0.53114$ (training), $R = 0.54541$ (validation), and $R = 0.52057$ (testing), resulting in an overall $R = 0.53171$. In the second dataset with random step sizes, we obtained $R = 0.57484$ (training), $R = 0.55223$ (validation), and $R = 0.55394$ (testing), leading to an overall R of 0.56834. The third dataset, with a sinusoidal signal, performed the best values, with $R = 0.73744$ (training), $R = 0.72915$ (validation), and $R = 0.67531$ (testing), resulting in an overall regression coefficient $R = 0.72444$. These results underscore the importance of tailoring the regression approach to the specific characteristics of each dataset. While the fixed step size and random step size datasets yielded distinct regression coefficients, the sinusoidal-signal-based dataset displayed the highest overall R values, highlighting the adaptability and efficacy of the FFBP-NN with the LM algorithm in handling diverse data types.

The comparative analysis of estimated joint angles from the system output and the FFBP-NN model is shown in Figure 8. We observe a remarkable congruence between the system outputs and the FFBP-NN model outputs in the first set of plots, representing the lowest MSE scenario with a random dataset. The blue and cyan lines represent the system and FFBP-NN model's outputs that appear nearly superimposed, indicating that the model accurately captures the underlying patterns in the random data. This suggests the model's performance is relatively acceptable in a scenario where the data is relatively unstructured and lacks a clear pattern. However, a different picture emerges in the second set of plots depicting the highest MSE with a sinusoidal dataset. Here, the system outputs (in blue) follow a distinct sinusoidal pattern, while the FFBP-NN model's outputs (in cyan) exhibit significant deviations. The model seems to struggle to accurately capture the cyclical nature of the sinusoidal data, resulting in a noticeable discrepancy between the two lines. This highlights the challenges that an FFBP-NN model may encounter when dealing with datasets characterized by complex, periodic, or oscillatory behaviors.

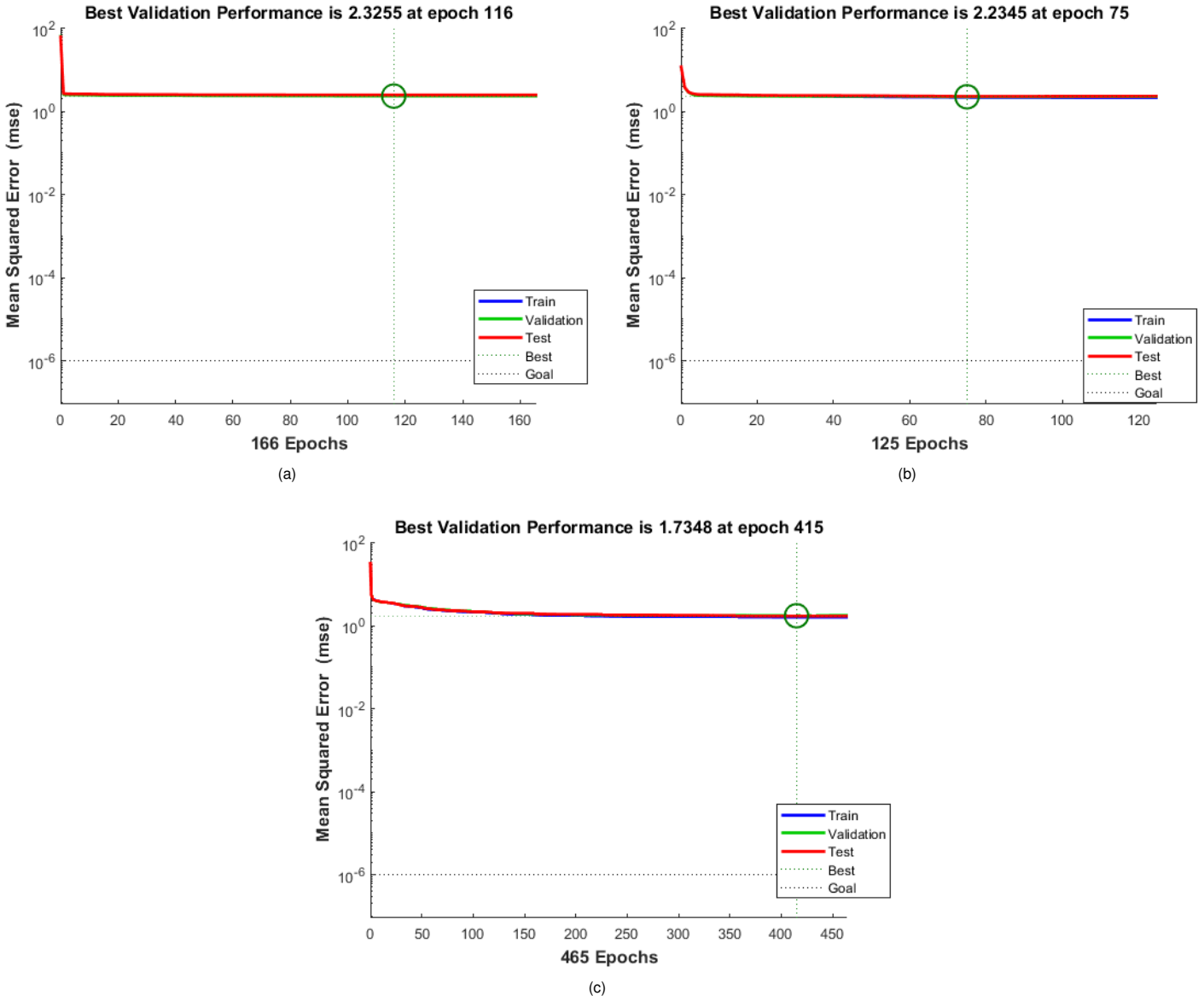
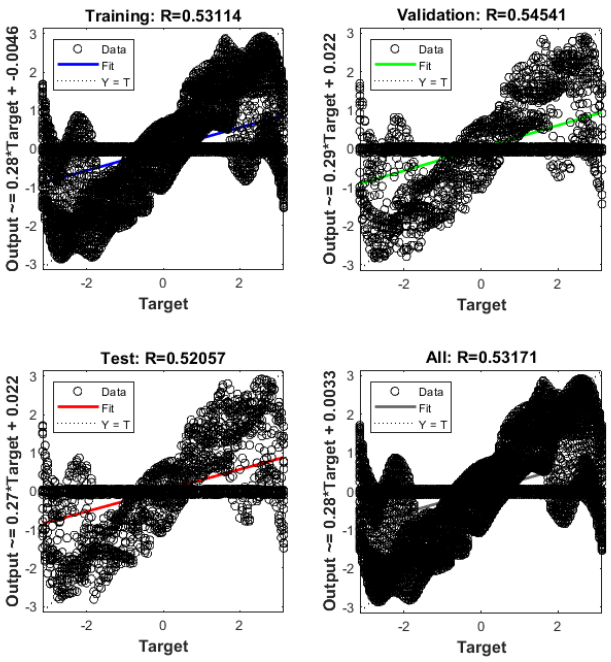


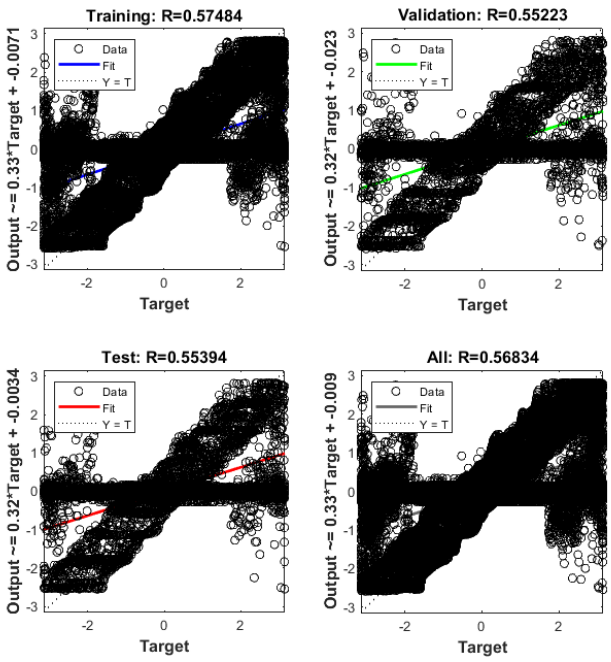
Figure 6 Lowest MSEs obtained with the LM algorithm and for the three different datasets: (a) Lowest MSE (Fixed Dataset, 1 hidden layer), (b) Lowest MSE (Random Dataset, 4 hidden layers), and (c) Lowest MSE (Sinusoidal Dataset, 4 hidden layers).

Table 3 Results of training with the LM algorithm for the three different proposed datasets.

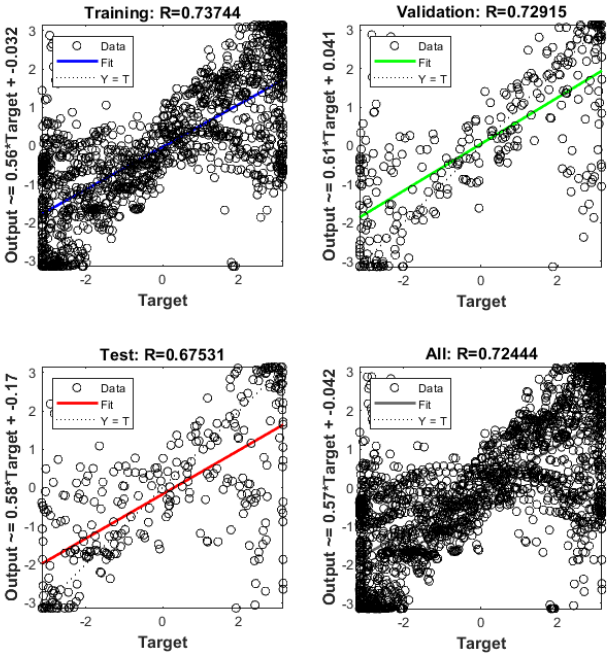
Number of hidden layers (number of neurons in each layer)	FIXED dataset	RANDOM dataset	SINUSOIDAL dataset
1 Hidden layer (10)	MSE: 2.3255 Epoch: 116	MSE: 2.2996 Epoch: 49	MSE: 3.6382 Epoch: 84
2 Hidden layers (10,2)	MSE: 2.3771 Epoch: 43	MSE: 2.2836 Epoch: 192	MSE: 3.1784 Epoch: 243
3 Hidden layers (10,20,2)	MSE: 2.4669 Epoch: 49	MSE:2.2813 Epoch: 78	MSE: 3.1859 Epoch: 29
4 Hidden layers (10,20,5,2)	MSE: 2.501 Epoch: 57	MSE: 2.2345 Epoch: 75	MSE: 2.4517 Epoch: 53
5 Hidden layers (10,20,10,5,2)	MSE: 2.5121 Epoch: 19	MSE: 2.3068 Epoch: 63	MSE: 2.9133 Epoch: 39



(a)



(b)



(c)

Figure 7 Regression results obtained by means of the LM algorithm and with the three proposed datasets: (a) Regression (Fixed Dataset, 1 hidden layer), (b) Regression (Random Dataset, 4 hidden layers), and (c) Regression (Sinusoidal Dataset, 4 hidden layers).

Feed-Forward Back-Propagation Neural Network with the Bayesian Regularization (BR) Algorithm

In the subsequent phase of our research, we examined the performance of the FFBP-NN, which had undergone training using the BR algorithm. This analysis encompassed an exploration of various datasets, including those of a fixed, random, and sinusoidal nature. The outcomes of this investigation illuminated pronounced variations in the network's performance, as visually represented in

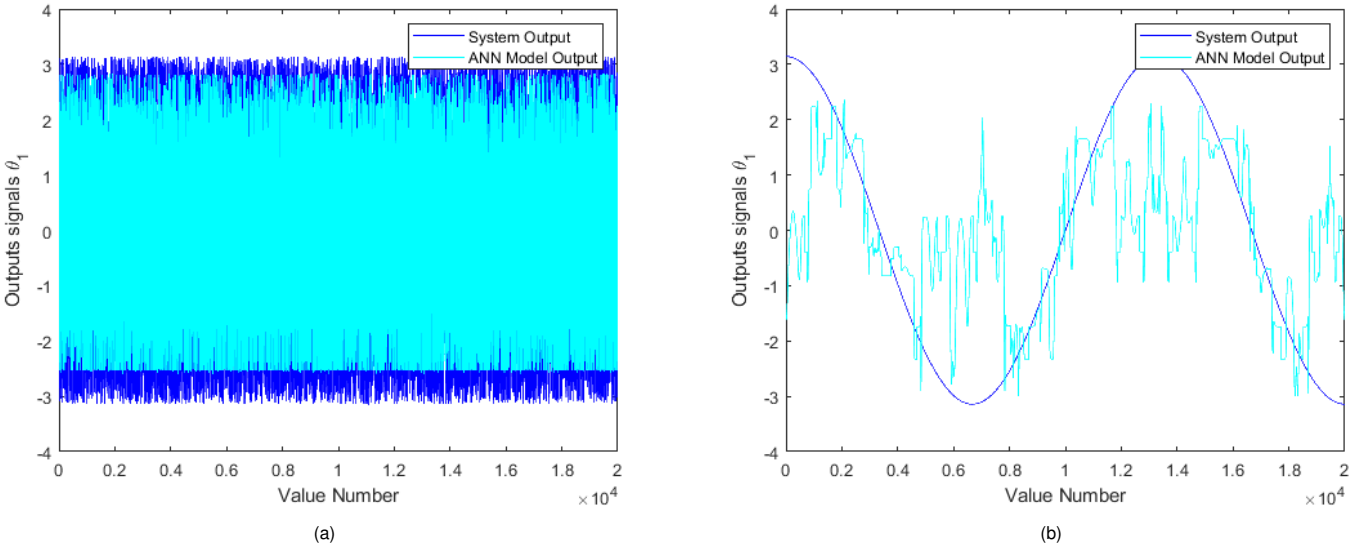


Figure 8 Comparative analysis of actual and estimated outputs obtained by means of the LM Algorithm: (a) Lowest MSE (Random Dataset, 4 hidden layers), and (b) Highest MSE (Sinusoidal Dataset, 1 hidden layer).

Figure 9. Notably, when the FFBP-NN underwent training on the fixed dataset, a remarkable achievement was observed, with the network achieving its most favorable MSE of 2.3348, a milestone reached precisely at epoch 37. The architectural configuration of this particular model featured three hidden layers with neuron configurations specified as (10, 20, 2). This outcome clearly demonstrated the BR algorithm's exceptional effectiveness in curbing overfitting tendencies and reinforcing the network's aptitude for generalization, resulting in notably proficient training outcomes when dealing with the fixed dataset. In a distinct vein, a different set of findings emerged when the FFBP-NN was subjected to training with random data. Specifically, it achieved its lowest recorded MSE, amounting to 2.1682, a noteworthy accomplishment at epoch 68. The architectural blueprint for this particular model diverged from the previous one, featuring a configuration of four hidden layers. This divergence in the network's architecture hinted at its adaptability and robustness, particularly in the face of more erratic and unpredictable data distributions, as with random data. Furthermore, our exploration extended to the training of the FFBP-NN on sinusoidal data. This endeavor yielded an MSE of 2.4632, attained at epoch 79. The architectural design for this specific scenario encompasses four hidden layers.

Our comprehensive comparison of these results underscores the multifaceted nature of the BR algorithm's impact on the FFBP-NN performance, as shown in Table 4. It facilitates the network's generalization and efficacy across various datasets with varying characteristics. The fixed dataset showcases the algorithm's proficiency in handling stable and non-random data patterns, whereas the random dataset demonstrates the network's adaptability to more unpredictable and erratic data distributions. Finally, the sinusoidal dataset reflects the algorithm's capability to address cyclical and periodic patterns, albeit with slightly lower performance than the fixed and random datasets. This nuanced understanding of the algorithm's behavior provides valuable insights for practitioners seeking to optimize neural network training across diverse datasets, emphasizing the significance of tailoring the approach to the specific data context.

When delving into the analysis of regression results for the BR algorithm, it becomes apparent that the primary focus lies on identifying the most optimal MSE value, particularly in the context of random data. The MSE is a pivotal metric in regression tasks, which is a crucial indicator of the model's accuracy in predicting target values. In the case of the BR algorithm, achieving the best possible MSE for random data sets a profound benchmark for the algorithm's performance. The consistent trend toward an MSE value close to 1 underscores the algorithm's capacity to minimize prediction errors and enhance the precision of its forecasts, as shown in Figure 10. Results showed promising performance in one experiment involving fixed step size data for training a three-layer FFBP-NN. The regression coefficient (R) during training reached the value 0.55022, indicating effective learning from the dataset. Validation yielded an R-value of 0.54578, demonstrating the network's generalization ability. Testing produced an R-value of 0.54834, reinforcing the model's real-world applicability. The overall R-value was 0.54922, indicating stability and accuracy in diverse scenarios. The results remained strong in a separate experiment using datasets with random step sizes and a four-layer neural network. The training phase achieved the R-value of 0.56858, showing adaptability to erratic data. Validation maintained high performance with an R of 0.57789. Testing, slightly lower at $R = 0.55223$, still offered reliability for practical use. The overall R-value was 0.56759, highlighting robustness even with challenging data. For a third dataset characterized by sinusoidal patterns, the FFBP-NN excelled. Training achieved an exceptional R of 0.70922, showing the model's ability to capture intricate patterns. Validation and testing displayed strong R-values (0.70532 and 0.70213), confirming suitability for sinusoidal data in real-world scenarios. The overall R-value of 0.70756 affirmed the network's robustness with complex, signal-based datasets.

In our training process, and as shown in Figure 11, we will delve deeper into the insights gained from our neural network model, using the BR algorithm. We will examine two distinct scenarios, each showcasing unique characteristics. In the first set of visual representations, we explore the domain of the lowest MSE. Here,

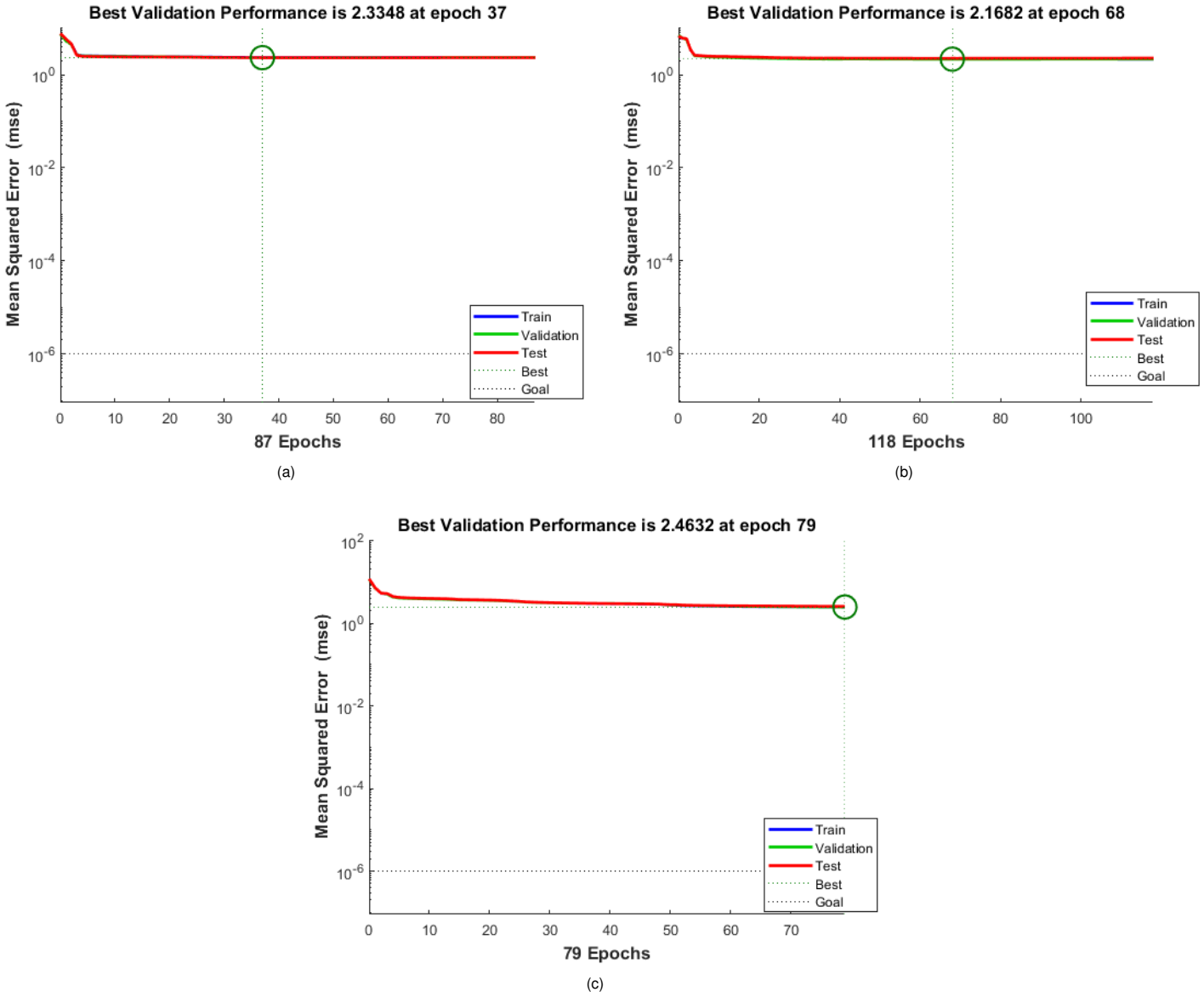


Figure 9 Lowest MSEs obtained using the BR algorithm and for the three generated datasets: (a) Lowest MSE (Fixed Dataset, 3 hidden layers), (b) Lowest MSE (Random Dataset, 4 hidden layers), and (c) Lowest MSE (Sinusoidal Dataset, 4 hidden layers).

Table 4 Results of training obtained with the BR algorithm for the three proposed different datasets.

Number of hidden layers (number of neurons in each layer)	FIXED dataset	RANDOM dataset	SINUSOIDAL dataset
1 Hidden layer (10)	MSE: 2.460 Epoch: 515	MSE: 2.3009 Epoch: 90	MSE: 3.4353 Epoch: 45
2 Hidden layers (10,2)	MSE: 2.4824 Epoch: 33	MSE: 2.1895 Epoch: 308	MSE: 3.1628 Epoch: 152
3 Hidden layers (10,20,2)	MSE: 2.3348 Epoch: 101	MSE: 2.2516 Epoch: 52	MSE: 2.7771 Epoch: 106
4 Hidden layers (10,20,5,2)	MSE: 2.3915 Epoch: 125	MSE: 2.1682 Epoch: 68	MSE: 2.4632 Epoch: 79
5 Hidden layers (10,20,10,5,2)	MSE: 2.4865 Epoch: 88	MSE: 2.3547 Epoch: 33	MSE: 2.8119 Epoch: 41

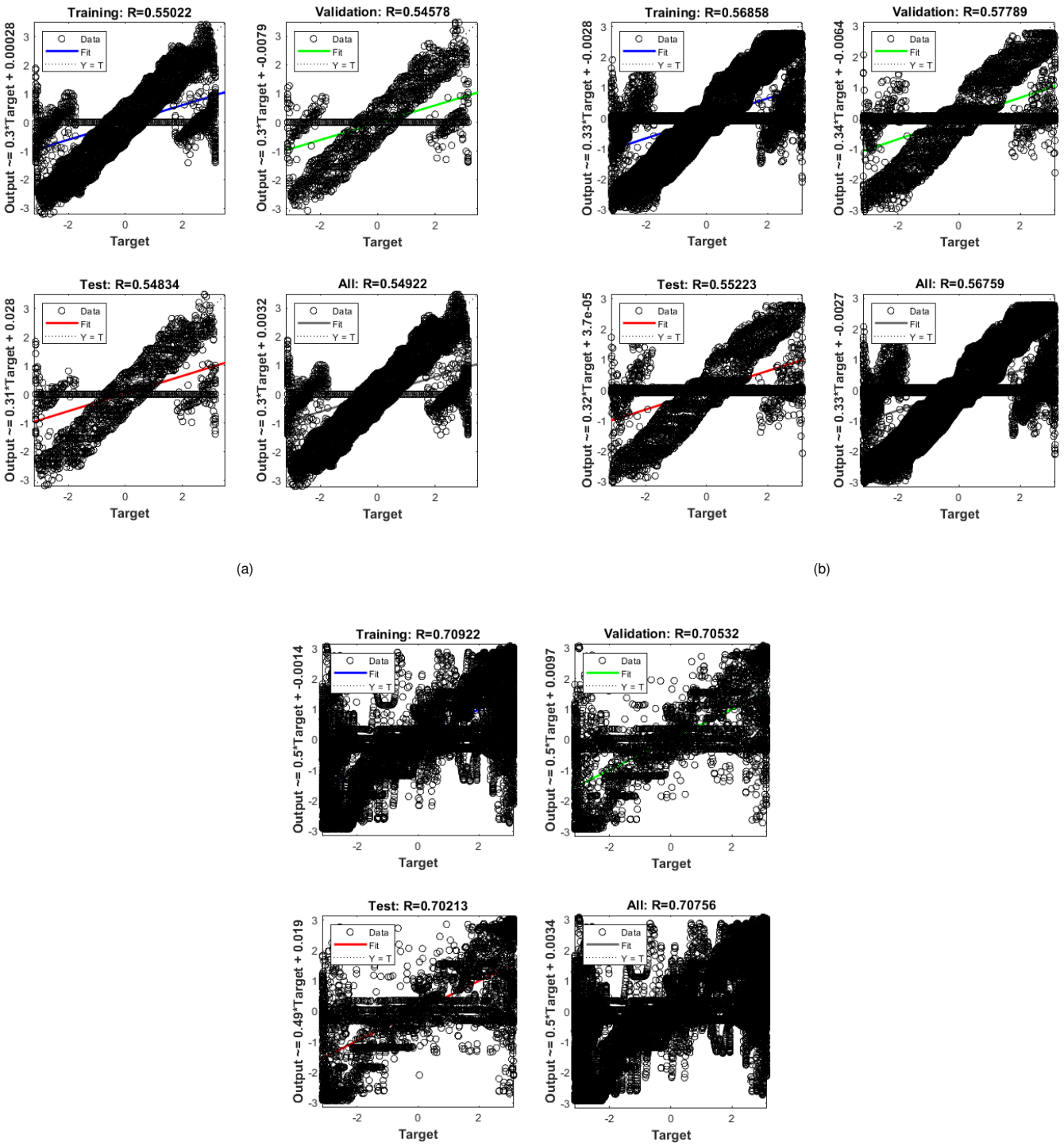


Figure 10 Regression results obtained with the BR algorithm and for the three distinct datasets: (a) Regression (Fixed Dataset, 3 hidden layers), (b) Regression (Random Dataset, 4 hidden layers), and (c) Regression (Sinusoidal Dataset, 4 hidden layers).

our neural network confronts a random dataset. It is immediately evident that there is a remarkable alignment between the system's outputs and the FFBP-NN model's outputs. The blue and cyan lines, representing the outputs, are almost identical, highlighting the model's proficiency in capturing the underlying complexity of the random data. This strong correspondence underscores the model's exceptional performance, especially when data lacks a clear structure or recognizable patterns. On the other hand, we

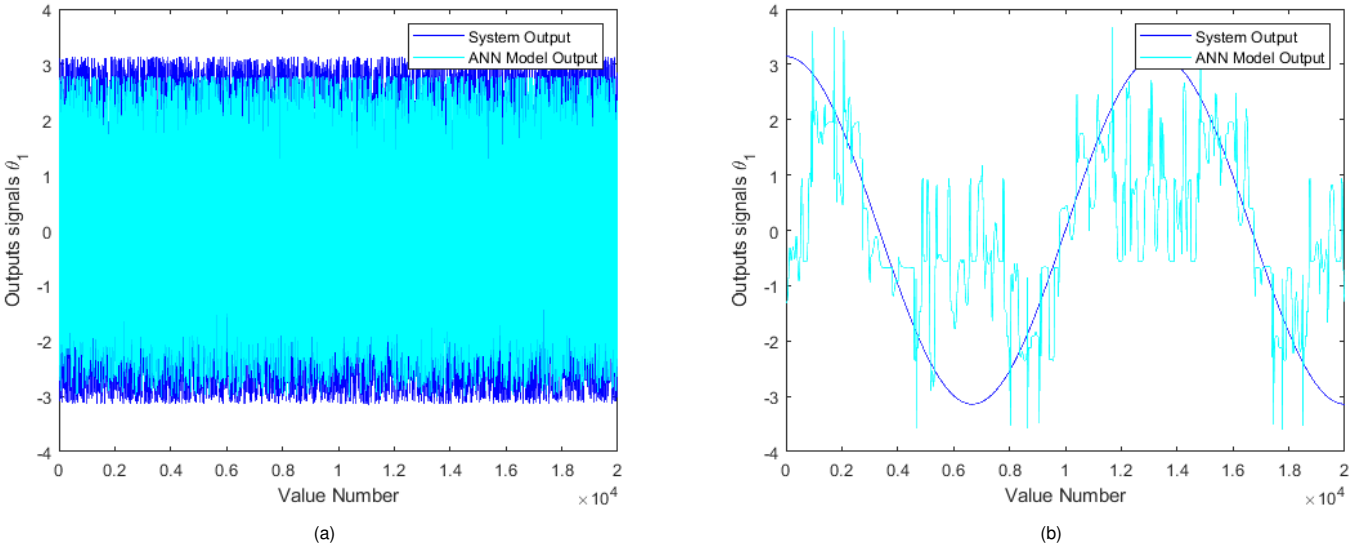


Figure 11 Comparative analysis of actual and estimated outputs obtained with the BR Algorithm and for two different datasets and for two different structures of the FFBP-NN model: (a) Lowest MSE (Random Dataset, 4 hidden layers), and (b) Highest MSE (Sinusoidal Dataset, 1 hidden layer).

shift our focus to the second set of visualizations, where we face the challenges of the highest MSE. In this case, our FFBP-NN model deals with a sinusoidal dataset. Here, the system's outputs, illustrated by the blue curve, gracefully follow the sinusoidal pattern, while the FFBP-NN model's outputs, depicted in cyan, exhibit noticeable deviations. It becomes apparent that the FFBP-NN model struggles to accurately replicate the cyclic nature inherent in sinusoidal data, resulting in significant disparities between the two curves.

Feed-Forward Back-Propagation Neural Network with the Scaled Conjugate Gradient (SCG) Algorithm

The SCG algorithm is highly regarded for its remarkable convergence rate and capability to effectively address intricate optimization challenges. In our investigation, we conducted a comparative analysis of training the FFBP-NN model using the SCG algorithm across different datasets: fixed, random, and sinusoidal. The outcomes exhibited notable variations in their performance, as depicted in Table 5. Specifically, when employing the SCG algorithm with the fixed data, the FFBP-NN model achieved its most favorable MSE of 2.4327 at the 130th epoch. The architectural configuration of this model comprised two hidden layers with respective neuron arrangements of (10, 2). Conversely, training the FFBP-NN model using random data yielded the best MSE of 2.2804 at the 463rd epoch, with a simpler model architecture consisting of 3 hidden layers configured as (10, 20, 2). Additionally, we conducted training on sinusoidal data, resulting in the FFBP-NN model achieving its lowest MSE of 2.4054 at the 835th epoch. The model architecture employed in this scenario also featured four hidden layers, as detailed in Figure 12. Some key observations emerge when comparing the FFBP-NN model's performance with the SCG algorithm across three datasets. The SCG algorithm excelled in optimizing performance on the fixed dataset, achieving a low MSE of 2.4327 at epoch 130, indicating its suitability for well-defined data. Conversely, on the random dataset, while still benefiting from SCG, it reached the lowest MSE of 2.2804 at epoch 463, indicating the challenge of adapting to random data. The SCG algorithm took longer for sinusoidal data to achieve its best MSE of 2.4054 at epoch 835, showing adaptability to different data types with consistent model architecture.

In summary, the SCG algorithm's efficacy in optimization tasks is underscored by its convergence rate, yet its performance is contingent on the characteristics of the dataset. It excels in scenarios with clear data patterns, such as fixed data, while it may require more time and iterations to converge on datasets with randomness or complex patterns, like random and sinusoidal data. Therefore, selecting the appropriate optimization algorithm and model architecture should be a deliberate decision based on the specific characteristics of the dataset and the desired outcomes.

In training FFBP-NN with the SCG algorithm, we conducted regression tasks on three distinct datasets to optimize the MSE. These experiments involved fixing the step size data and utilizing two hidden layers. The results of the first dataset revealed regression coefficients as follows: an R-value of 0.49759 for the training set, an R-value of 0.5263 for the validation set, an R-value of 0.49396 for the test set, and an overall R-value of 0.50141. In the second dataset, we focused on minimizing the MSE using random step size data and expanding to three hidden layers. The corresponding regression statistics were found to be an R-value of 0.55522 for the training set, an R-value of 0.55847 for the validation set, an R-value of 0.55722 for the test set, and an overall R-value of 0.55599. Lastly, when dealing with a dataset based on a sinusoidal signal, the aim was again to minimize the MSE. The regression results for this dataset included an R-value of 0.72855 for the training set, an R-value of 0.7177 for the validation set, an R-value of 0.72493 for the test set, and an overall R-value of 0.72638. These distinct regression analyses shed light on the varying performance of the FFBP-NN trained with the SCG algorithm across different datasets, step sizes, and hidden layer configurations, offering valuable insights for further optimization and model selection, as shown in Figure 13.

A comparative examination was conducted to assess the actual

Table 5 Results obtained after training different FFBP-NN models with the SCG algorithm and for the three different datasets.

Number of hidden layers (number of neurons in each layer)	FIXED dataset	RANDOM dataset	SINUSOIDAL dataset
1 Hidden layer (10)	MSE: 2.5814 Epoch: 158	MSE: 2.4924 Epoch: 90	MSE: 3.7088 Epoch: 537
2 Hidden layers (10,2)	MSE: 2.4327 Epoch: 130	MSE: 2.3492 Epoch: 407	MSE: 3.6975 Epoch: 606
3 Hidden layers (10,20,2)	MSE: 2.5152 Epoch: 188	MSE: 2.2804 Epoch: 463	MSE: 2.6966 Epoch: 998
4 Hidden layers (10,20,5,2)	MSE: 2.4944 Epoch: 269	MSE: 2.2983 Epoch: 213	MSE: 2.4054 Epoch: 835
5 Hidden layers (10,20,10,5,2)	MSE: 2.4609 Epoch: 146	MSE: 2.349 Epoch: 577	MSE: 3.4659 Epoch: 384

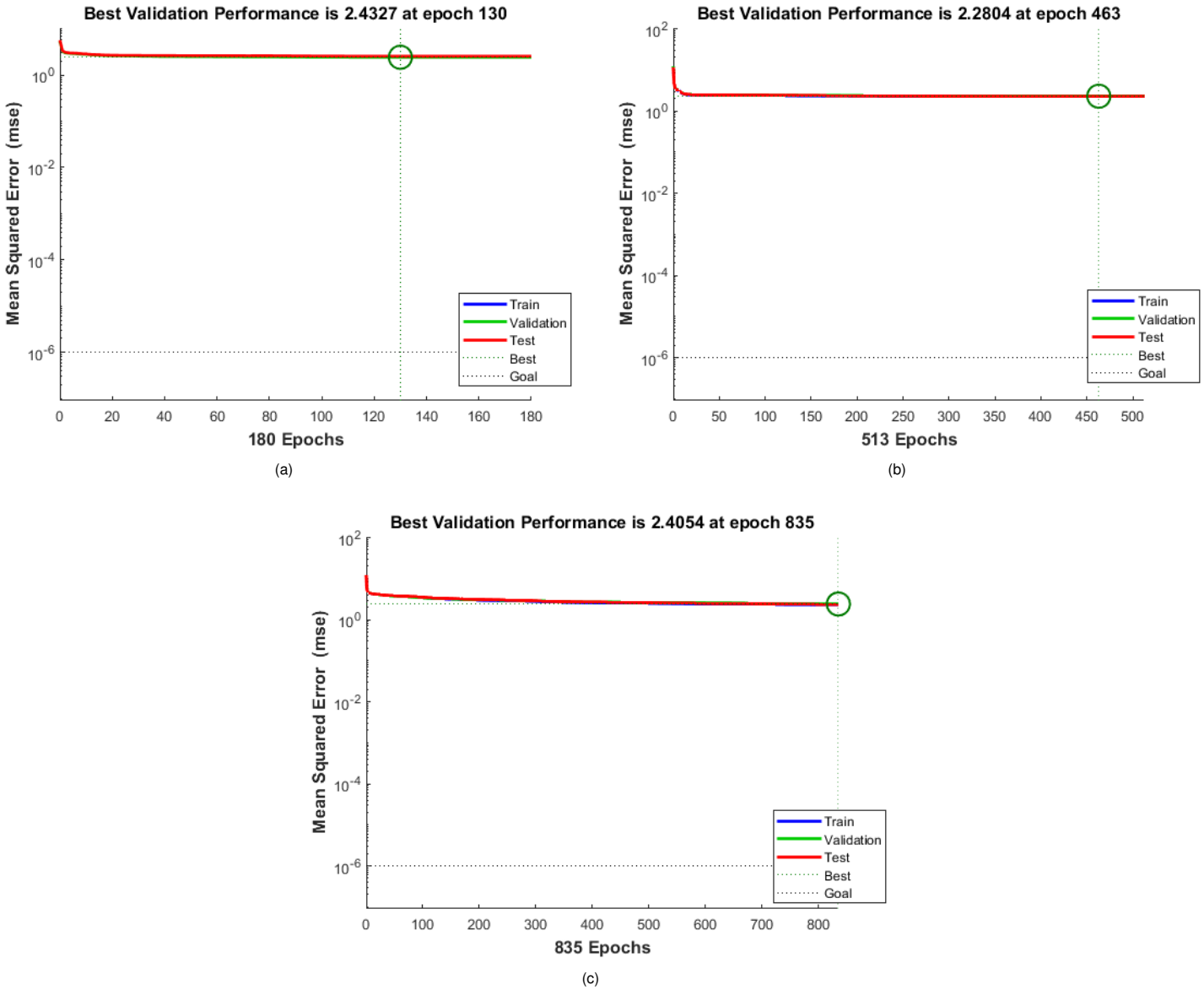


Figure 12 Lowest MSEs obtained by training the FFBP-NN models with the SCG algorithm and for the three different datasets: (a) Lowest MSE (Fixed Dataset, 2 hidden layers), (b) Lowest MSE (Random Dataset, 3 hidden layers), and (c) Lowest MSE (Sinusoidal Dataset, 4 hidden layers).

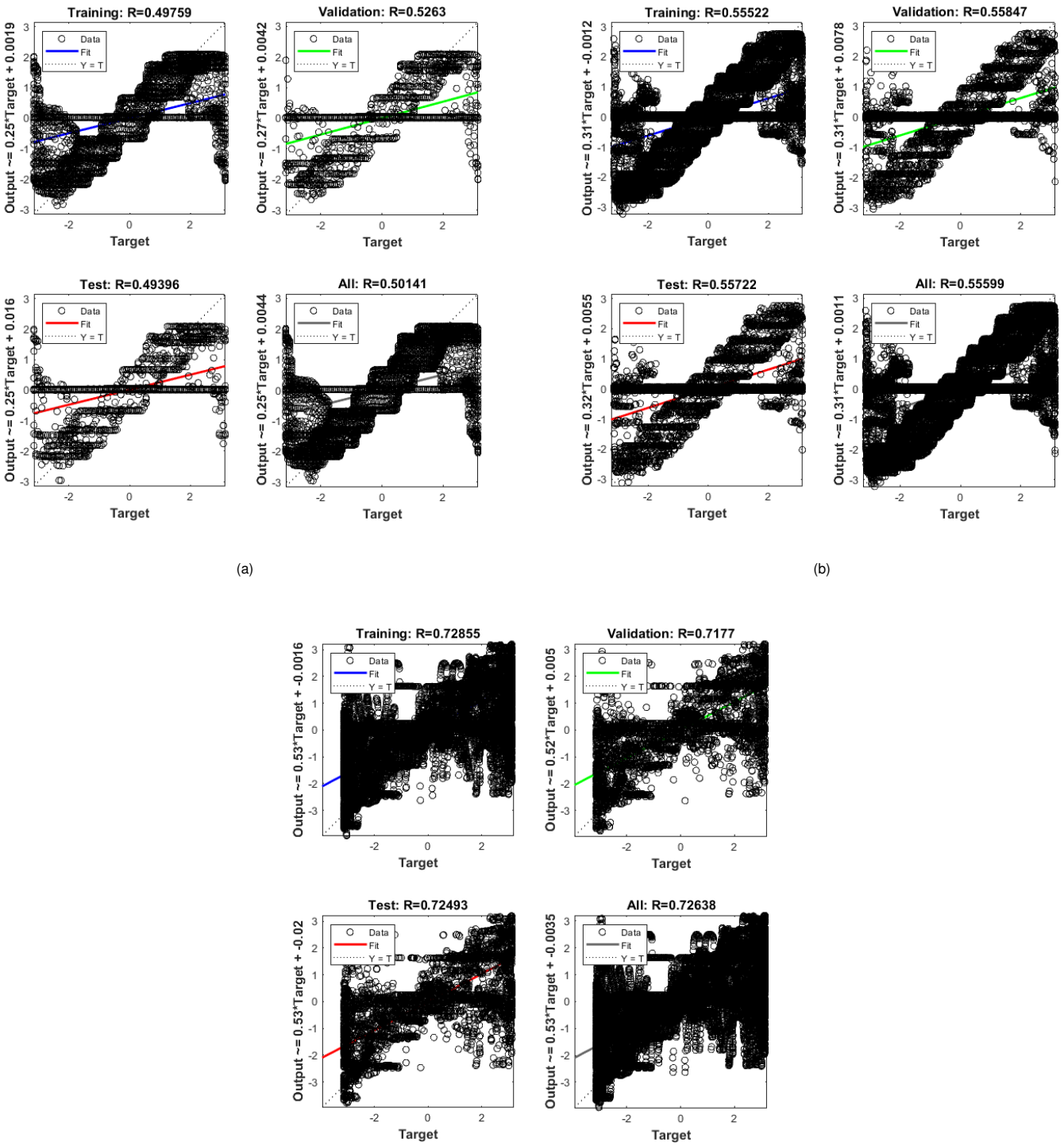


Figure 13 Regression results of some proposed FFBP-NN models obtained with the SCG algorithm and for the three different datasets: (a) Regression (Fixed Dataset, 2 hidden layers), (b) Regression (Random Dataset, 3 hidden layers), and (c) Regression (Sinusoidal Dataset, 4 hidden layers).

and predicted results produced by the SCG Algorithm using two different datasets. When employing the SCG algorithm, it became evident that our neural network excelled when dealing with random data. However, it faced difficulties in accurately reproducing sinusoidal patterns. The analysis comparing the estimated joint angles between the system output and the FFBP-NN model is illus-

trated in Figure 14. In the first series of plots, which represents the scenario with the lowest MSE using a random dataset, a striking similarity is observed between the outputs of the system and the FFBP-NN model. The blue and cyan lines, representing the system and FFBP-NN model outputs, closely overlap, indicating that the model effectively captures the underlying patterns within the random data. This suggests the model performs reasonably well when the data lacks a clear structure and is relatively unorganized. Conversely, a different story unfolds in the second series of plots depicting the scenario with the highest MSE using a sinusoidal dataset. In this case, the system outputs (depicted in blue) follow a clear sinusoidal pattern, while the FFBP-NN model's outputs (depicted in cyan) exhibit noticeable deviations. The model appears to encounter challenges in accurately replicating the cyclical nature of sinusoidal data, resulting in a significant disparity between the two lines. This emphasizes the difficulties of an FFBP-NN model when confronted with datasets characterized by intricate, periodic, or oscillatory behaviors.

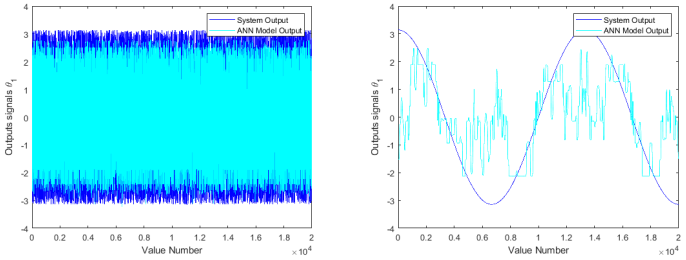


Figure 14 Comparative analysis of actual and estimated outputs obtained with the SCG Algorithm and using two different datasets: (a) Lowest MSE (Random Dataset, 3 hidden layers), and (b) Highest MSE (Sinusoidal Dataset, 1 hidden layer).

CONCLUSION

In conclusion, our work presented a contribution in the fields of robotics and automation through the development and implementation of various Feed-Forward Back-Propagation Neural Network (FFBP-NN) algorithms for the computation of the inverse kinematics of a 2-DoF manipulator robot. A pivotal aspect of our research involves an in-depth exploration of these algorithms, coupled with rigorous experimentation and evaluations to discern their performance characteristics.

We mainly used the MSE metric to evaluate the performance of the different proposed neural network architectures. Based on such metric, our findings highlighted the substantial impact of training with Levenberg-Marquardt (LM) and Bayesian Regularization (BR) algorithms, particularly noting that the optimal results (that correspond to the lowest MSEs) were achieved when trained with random-step-size datasets in the context of a four-hidden-layer configuration. Similarly, for the Scaled Conjugate Gradient (SCG) algorithm, we observed the best outcomes (lowest MSE) when employing random-step-size datasets in a three-hidden-layer setting. This nuanced understanding of algorithmic behavior provides valuable insights for practitioners. Most notably, our work achieves a remarkable milestone by applying FFBP-NN models to address inverse kinematics problems with an unprecedented level of precision and reliability, surpassing the capabilities of traditional methods. These outcomes underscore the transformative potential of FFBP-NN models in tackling complex problems within the realm of robotics.

In future works, it is essential to expand the applicability of the proposed FFBP-NN models by testing them on manipulator robots with greater degrees of freedom, such as SCARA or 6-DoF industrial robots. This evaluation will assess the scalability and adaptability of the FFBP-NN models across diverse robotic platforms.

Furthermore, in order to improve the selection phase of the hyperparameters of the FFBP-NN, the idea is to integrate the metaheuristic optimization algorithms within the training process in order to find the optimal architecture to provide the best accuracy or the lowest MSE. In addition, the objective is to explore and utilize more advanced ANN architectures, such as Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), for the modeling of the inverse kinematics of manipulator robots. This exploration can significantly enhance the capabilities of the training process and prediction when operating with a complex structure of the robot and also with a high number of degrees of freedom and then for redundant manipulator robots.

Moreover, the current work can be extended and applied to the control part of manipulator robots using some nonlinear control approaches as those proposed in (Jenhani *et al.* 2022) for the position control of robotic systems, as well as for particular applications like in medical robotics such as exoskeleton systems for pediatric gaits (Narayan *et al.* 2018, 2023).

Availability of data and material

The data used to support the findings of the study are included within the article.

Conflicts of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Ethical standard

The authors have no relevant financial or non-financial interests to disclose.

LITERATURE CITED

Abbas, M., J. Narayan, and S. K. Dwivedy, 2019 Simulation analysis for trajectory tracking control of 5-DOFs robotic arm using ANFIS approach. In *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pp. 1–6.

Aravinddhakshan, S., S. Apte, and S. M. Akash, 2021 Neural network based inverse kinematic solution of a 5 DOF manipulator for industrial application. Journal of Physics: Conference Series **1969**: 012010.

Aysal, F. E., İ. Çelik, E. Cengiz, and Y. Oğuz, 2023 A comparison of multi-layer perceptron and inverse kinematic for RRR robotic arm. Politeknik Dergisi pp. 1–1.

Becerra, G. and R. Kremer, 2011 Ambient intelligent environments and environmental decisions via agent-based systems. Journal of Ambient Intelligence and Humanized Computing **2**: 185–200.

Benavente-Peces, C., A. Ahrens, and J. Filipe, 2014 Advances in technologies and techniques for ambient intelligence.

Bouzid, R., J. Narayan, and H. Gritli, 2023 Feedforward backpropagation artificial neural network for modeling the forward kinematics of a robotic manipulator. In *2023 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, pp. 302–307, Sakheer, Bahrain.

Bouzid, R., J. Narayan, and H. Gritli, 2024a Artificial neural networks for the forward kinematics of a SCARA manipulator: A comparative study with two datasets. In *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, pp. 1792–1797.

Bouzid, R., J. Narayan, and H. Gritli, 2024b Exploring neural networks for forward kinematics of the robotic arm with different length configurations: A comparative analysis. In *2024 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, volume 2, pp. 1–6.

Bouzid, R., J. Narayan, and H. Gritli, 2024c Investigating neural network hyperparameter variations in robotic arm inverse kinematics for different arm lengths. In *2024 Third International Conference on Power, Control and Computing Technologies (ICPC2T)*, pp. 351–356.

Bouzid, R., J. Narayan, and H. Gritli, 2024d Solving inverse kinematics problem for manipulator robots using artificial neural network with varied dataset formats. In *Complex Systems and Their Applications*, edited by E. Campos-Cantón, G. Huerta-Cuellar, E. Zambrano-Serrano, and E. Tlelo-Cuautle, pp. 55–78, Cham, Springer Nature Switzerland.

Cagigas-Muñiz, D., 2023 Artificial neural networks for inverse kinematics problem in articulated robots. Engineering Applications of Artificial Intelligence **126**: 107175.

Cimen, M. E., Z. Garip, M. A. Pala, A. F. Boz, and A. Akgül, 2019 Modelling of a chaotic system motion in video with artificial neural networks. Chaos Theory and Applications **1**: 38 – 50.

Darba, A., N. B. Sushmi, and D. Subbulekshmi, 2022 Performance analysis of FFBP-LM-ANN based hourly GHI prediction using environmental variables: A case study in chennai. Mathematical Problems in Engineering **2022**: 1713657.

Dash, K. K., B. B. Choudury, and S. K. Senapati, 2017 A inverse kinematic solution of a 6-DOF industrial robot using ANN. Indian Journal of Scientific Research **15**: 97–101.

del Rosario Martinez-Blanco, M., V. H. Castañeda-Miranda, G. Ornelas-Vargas, H. A. Guerrero-Osuna, L. O. Solis-Sanchez, *et al.*, 2016 Generalized regression neural networks with application in neutron spectrometry. In *Artificial Neural Networks*, edited by J. L. G. Rosa, chapter 3, IntechOpen, Rijeka.

Denavit, J. and R. S. Hartenberg, 1955 A kinematic notation for lower-pair mechanisms based on matrices. Journal of Applied Mechanics **22**: 215–221.

Di Pietro, A., D. Torresi, M. Zadro, L. Cosentino, C. Ducoin, *et al.*, 2012 The inverse kinematics thick target scattering method as a tool to study cluster states in exotic nuclei. Journal of Physics: Conference Series **366**: 012013.

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

Dumitriu, D. N., O. D. Melinte, and M. Ionescu, 2020 Neural networks kinematics guidance of lewansoul learm 5r serial manipulator. Acta Electrotehnica **61**.

Ganapathy, S., 1984 Decomposition of transformation matrices for robot vision. Pattern Recognition Letters **2**: 401–412.

Gao, B., Z. Zhu, J. Zhao, and L. Jiang, 2017 Inverse kinematics and workspace analysis of a 3 DOF flexible parallel humanoid neck robot. Journal of Intelligent & Robotic Systems **87**: 211–229.

Gao, R., 2020 Inverse kinematics solution of robotics based on neural network algorithms. Journal of Ambient Intelligence and Humanized Computing **11**: 6199–6209.

García-Samartín, J. F. and A. Barrientos, 2023 Kinematic modelling of a 3RRR planar parallel robot using genetic algorithms and neural networks. Machines **11**.

Ghaleb, N. M. and A. A. Aly, 2018 Modeling and control of 2-DOF robot arm. International Journal of Emerging Engineering Research and Technology **6**: 24–31.

Handayani, A. N., N. Lathifah, H. W. Herwanto, R. A. Asmara, and K. Arai, 2018 Neural network bayesian regularization backpropagation to solve inverse kinematics on planar manipulator. In *2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, pp. 99–104, IEEE.

Huo, L. and L. Baron, 2008 The joint-limits and singularity avoidance in robotic welding. Industrial Robot: An International Journal **35**: 456–464.

Ibarra-Pérez, T., J. M. Ortiz-Rodríguez, F. Olivera-Domingo, H. A. Guerrero-Osuna, H. Gamboa-Rosales, *et al.*, 2022 A novel inverse kinematic solution of a six-DOF robot using neural networks based on the taguchi optimization technique. Applied Sciences **12**: 9512.

Jenhani, S., H. Gritli, and G. Carbone, 2022 Comparison between some nonlinear controllers for the position control of Lagrangian-type robotic systems. Chaos Theory and Applications **4**: 179 – 196.

Karaca, Y., 2023 Computational complexity-based fractional-order neural network models for the diagnostic treatments and predictive transdifferentiability of heterogeneous cancer cell propensity. Chaos Theory and Applications **5**: 34 – 51.

Kayri, M., 2016 Predictive abilities of Bayesian regularization and Levenberg–Marquardt algorithms in artificial neural networks: a comparative empirical study on social data. Mathematical and Computational Applications **21**: 20.

Keleş, Z., G. Sonugür, and M. Alçın, 2023 The modeling of the rucklidge chaotic system with artificial neural networks. Chaos Theory and Applications **5**: 59 – 64.

Kim, J. S., Y. H. Jeong, and J. H. Park, 2016 A geometric approach for forward kinematics analysis of a 3-sps/s redundant motion manipulator with an extra sensor using conformal geometric algebra. Meccanica **51**: 2289–2304.

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

Kumar, P. *et al.*, 2018 Artificial neural network based geometric error correction model for enhancing positioning accuracy of a robotic sewing manipulator. Procedia Computer Science **133**: 1048–1055.

Lathifah, N., A. N. Handayani, H. W. Herwanto, and S. Sendari, 2018 Solving inverse kinematics trajectory tracking of planar manipulator using neural network. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pp. 483–488, IEEE.

Li, H. and A. V. Savkin, 2018 An algorithm for safe navigation of mobile robots by a sensor network in dynamic cluttered industrial environments. Robotics and Computer-Integrated Manufacturing **54**: 65–82.

Liu, W., D. Chen, and J. Steil, 2017 Analytical inverse kinematics solver for anthropomorphic 7-DOF redundant manipulators with human-like configuration constraints. Journal of Intelligent & Robotic Systems **86**: 63–79.

Madhuraghava, P., B. D. Fakruddin, R. V. Subhash, and N. Sunil, 2018 Modelling and structural, analysis of a 6-DOF robot spray coating manipulator. The International Journal of Engineering and Science **7**: 48–56.

Mahajan, A., H. Singh, and N. Sukavanam, 2017 An unsupervised learning based neural network approach for a robotic manipulator. International Journal of Information Technology **9**: 1–6.

Martinez-garcia, J. A., A. M. Gonzalez-zapata, E. J. Rechy-ramirez, and E. Tlelo-cuautle, 2022 On the prediction of chaotic time series using neural networks. Chaos Theory and Applications **4**: 94 – 103.

Møller, M. F., 1993 A scaled conjugate gradient algorithm for fast supervised learning. Neural networks **6**: 525–533.

Narayan, J., M. Abbas, B. Patel, and S. K. Dwivedy, 2023 Adaptive RBF neural network-computed torque control for a pediatric gait exoskeleton system: an experimental study. Intelligent Service Robotics **232**: 726–732.

Narayan, J., S. Banerjee, D. Kamireddy, and S. K. Dwivedy, 2022 Fuzzy membership functions in ANFIS for kinematic modeling of 3R manipulator. In *Handbook of Smart Materials, Technologies, and Devices: Applications of Industry 4.0*, edited by C. M. Hussain and P. Di Sia, pp. 1101 – 119, Springer International Publishing, Cham.

Narayan, J. and A. Singla, 2017a ANFIS based kinematic analysis of a 4-DOFs SCARA robot. In *2017 4th International Conference on Signal Processing, Computing and Control (ISPCC)*, pp. 205–211.

Narayan, J. and A. G. Singla, 2017b *Inverse Kinematic Study of Spatial Serial Manipulators using ANFIS Approach*. Ph.D. thesis, Thapar Institute of Engineering and Technology.

Narayan, J., E. Singla, S. Soni, and A. Singla, 2018 Adaptive neuro-fuzzy inference system–based path planning of 5-degrees-of-freedom spatial manipulator for medical applications. Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine **232**: 726–732.

Noorani, I. and F. Mehrdoust, 2022 Parameter estimation of uncertain differential equation by implementing an optimized artificial neural network. Chaos, Solitons & Fractals **165**: 112769.

Petrescu, R. V., R. Aversa, B. Akash, R. Bucinell, J. Corchado, *et al.*, 2017 Inverse kinematics at the anthropomorphic robots, by a trigonometric method. American Journal of Engineering and Applied Sciences **10**: 394–411.

Petrović, L., 2018 Motion planning in high-dimensional spaces. arXiv preprint arXiv:1806.07457 .

Ranganathan, A., 2004 The levenberg-marquardt algorithm. Tutoral on LM algorithm **11**: 101–110.

Rea Minango, S. N. and J. C. E. Ferreira, 2017 Combining the step-nc standard and forward and inverse kinematics methods for generating manufacturing tool paths for serial and hybrid robots. International Journal of Computer Integrated Manufacturing **30**: 1203–1223.

Reiter, A., A. Müller, and H. Gattringer, 2018 On higher order inverse kinematics methods in time-optimal trajectory planning for kinematically redundant manipulators. IEEE Transactions on Industrial Informatics **14**: 1681–1690.

Snieder, R., 1998 The role of nonlinearity in inverse problems. Inverse problems **14**: 387.

Takatani, H., N. Araki, T. Sato, and Y. Konishi, 2019 Neural network-based construction of inverse kinematics model for serial redundant manipulators. Artificial Life and Robotics **24**: 487–493.

Theofanidis, M., S. I. Sayed, J. Cloud, J. Brady, and F. Makedon, 2018 Kinematic estimation with neural networks for robotic manipulators. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part III 27*, pp. 795–802, Springer.

Wagaa, N., H. Kallel, and N. Mellouli, 2023 Analytical and deep learning approaches for solving the inverse kinematic problem of a high degrees of freedom robotic arm. Engineering Applications of Artificial Intelligence **123**: 106301.

Wang, W., G. Yu, M. Xu, and D. Walker, 2014 Coordinate transformation of an industrial robot and its application in deterministic optical polishing. Optical Engineering **53**: 055102–055102.

Xu, W., Z. Mu, T. Liu, and B. Liang, 2017 A modified modal method for solving the mission-oriented inverse kinematics of hyper-redundant space manipulators for on-orbit servicing. Acta Astronautica **139**: 54–66.

Zhao, D., Y. Bi, and Y. Ke, 2018 Kinematic modeling and inverse kinematics solution of a new six-axis machine tool for oval hole drilling in aircraft wing assembly. The International Journal of Advanced Manufacturing Technology **96**: 2231–2243.

How to cite this article: Bouzid, R., Gritli, H., and Narayan, J. Investigating Feed-Forward Back-Propagation Neural Network with Different Hyperparameters for Inverse Kinematics of a 2-DoF Robotic Manipulator: A Comparative Study *Chaos Theory and Applications*, 6(2), 90-110, 2024.