machines

MDPI

Article

A Novel Hybrid IK Architecture for Robotic Arms: Iterative Refinement of Soft-Computing Approximations with Validation on ABB IRB-1200 Robotic Arm

Meenalochani Jayabalan [1,*], Karunamoorthy Loganathan [2] and Palanikumar Kayaroganam [3]

[1] Research Scholar, Department of Mechanical Engineering, College of Engineering Guindy, Anna University, Chennai 600025, India
[2] Department of Mechanical Engineering, College of Engineering Guindy, Anna University, Chennai 600025, India
[3] Department of Mechanical Engineering, Sri Sai Ram Institute of Technology, Chennai 6000132, India
* Correspondence: meenakarthik.ece@gmail.com

Abstract

Adaptive Neuro-Fuzzy Inference System (ANFIS)-based inverse kinematics (IK) is highly accurate for trained poses but often yields approximations for unseen inputs due to non-standardized training data. This research addresses these limitations through two novel contributions designed for any generic Degrees of Freedom (DoF) serial revolute robotic arm. First, A structured training methodology is introduced using workspace decomposition and cubic path planning. Instead of random sampling, the workspace is partitioned into cubic regions where 28 unique trajectories (12 edges, 12 face diagonals, four space diagonals) connect the eight vertices using cubic polynomial interpolation. This ensures physically consistent data mirroring real world point to point (PTP) movements. Even though validated on an ABB IRB-1200 robotic arm, this modular design is inherently scalable, allowing the local cubic expertise to be extended to cover the entire reachable workspace. Second, a two-stage hybrid IK framework is proposed, where an initial ANFIS approximation is refined via Jacobian-based iterative methods. Three Hybrid Frame works were evaluated, Framework-1 (ANFIS + Jacobian Gradient), Framework-2 (ANFIS + Jacobian Pseudoinverse/Newton–Raphson), and Framework-3 (ANFIS + Damped Least Squares). The results show that all three hybrid IK frameworks achieve reliable convergence, while the DLS-based hybrid provides the best trade-off between accuracy, convergence speed, and numerical stability. This generic, analytical free architecture provides a computationally efficient solution even in a hybrid scenario, bridging the gap between offline structured training and online, real-time refinement for digital twin synchronization and industrial automation.

Keywords: robotic arm; 6-DoF; N-DoF; hybrid inverse kinematics; soft computing; iterative IK; hybrid IK; ANFIS; gradient IK method; pseudoinverse Newton–Raphson; damped least squares (DLS); ABB IRB 1200; dataset generation

Check for updates

Received: 1 January 2026
Revised: 20 February 2026
Accepted: 20 February 2026
Published: 4 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

To execute tasks in three-dimensional space, a robotic arm must solve a fundamental geometric problem: determining the joint angles that place its end-effector at a desired position and orientation. This is the inverse kinematics (IK) problem. While forward kinematics (FK) provides a unique mapping from joint variables to end-effector pose,

the inverse mapping is nonlinear and often admits multiple or no valid solutions. This inherent complexity makes IK a central challenge in robot motion planning and control, motivating a wide range of solution approaches, from closed-form analytical models to modern learning-based algorithms.

1.1. Literature Review

Analytical IK aims to derive closed-form equations that directly compute joint variables from a desired end-effector pose. Early work established that closed-form IK is feasible for non-redundant serial manipulators, particularly classical 6-DOF industrial robots [1]. Geometric solutions have been applied to humanoid arms and collaborative robots, achieving accurate results under structured kinematic assumptions [2]. More recent advancements include solutions based on conformal geometric algebra for 7-DOF manipulators [3] and closed-form solutions for collaborative robots with non-spherical wrists [4]. Unified analytical–numerical formulations using quaternion mathematics have also been reported to improve modeling accuracy [5]. Although analytical methods are computationally efficient, they depend heavily on robot geometry. Even minor offsets, redundancies, or non-standard Denavit–Hartenberg parameters make closed-form derivation extremely difficult or impossible, limiting their applicability to modern manipulators.

Numerical IK emerged as a general alternative to rigid analytical formulations. Foundational iterative strategies include the Newton–Raphson method for IK updates [6], the Jacobian Pseudoinverse method for redundant manipulators [7], and the Damped Least Squares (DLS) technique, which improves stability near singularities [8]. These methods provide general, widely applicable solutions and have been employed in real-time implementations [9], as well as in redundant robot singularity analysis [10]. However, numerical IK methods depend on good initial guesses, and their convergence may degrade in low-manipulability or near-singularity regions. They also provide only local solutions. Heuristic IK methods were designed primarily for computer animation, prioritizing fast, lightweight computation over strict physical accuracy. FABRIK, a widely used method, performs IK through forward–backward reaching without computing Jacobians [11]. Other lightweight approaches include Gauss–Seidel-based solvers and Cyclic Coordinate Descent (CCD) [12], which have been well documented in animation literature [13]. Yet, these methods cannot maintain the precision, stability, and constraint handling required for industrial robotic manipulation. Comparative work confirms that neural and heuristic solvers perform well for animation but fail to meet robotic accuracy requirements [14]. Optimization and metaheuristic IK methods allow flexible handling of joint limits, collision avoidance, and multi-objective constraints. Notable contributions include IK solutions using BODE-CS [15], improved Bald Eagle Search optimization [16], SQP/BP-SQP solvers for industrial robots [17], and general metaheuristic IK solution strategies [18]. PSO combined with ANN has been applied successfully to redundant manipulators [19]. Despite their flexibility, these methods are computationally expensive and not suitable for real-time applications where IK must be solved at high frequency.

Learning-based techniques approximate IK without requiring explicit kinematic modelling. Early contributions include VQTAM-based manipulator learning [20], neural IK models for anthropomorphic robots [21], reinforcement-learning approaches for industrial manipulation [22], Taguchi-optimized neural IK [23], sequential machine-learning IK solvers [24], FOA-optimized networks [25], and ANN posture-control strategies [26]. ANFIS-based IK has also been widely explored, with successful demonstrations on PUMA manipulators, 6-DOF robot arms, and SCARA robots [27–30]. These models offer fast inference but depend heavily on dataset quality. Poor training coverage leads to weak

generalization in unsampled regions, a well-known limitation across soft-computing IK studies.

Hybrid IK strategies integrate machine-learning approximation with numerical refinement to combine speed and accuracy. Early hybrid attempts paired neural models with Jacobian-based solvers for local correction. Recent works include hybrid Differential Evolution + Jacobian Pseudoinverse IK [31], which demonstrated improved convergence. However, most hybrid IK studies still rely on randomly sampled training data, which fails to uniformly represent the workspace. Additionally, existing works rarely compare multiple hybrid refinement strategies under identical conditions. These gaps highlight the need for hybrid IK frameworks supported by structured dataset generation and consistent refinement evaluation, which forms the core objective of the present work.

1.2. Research Gaps Identified

Despite the large amount of research on inverse kinematics, several important gaps still exist. ANFIS-based IK performance strongly depends on the quality and coverage of the training data, and accuracy decreases in workspace regions that are poorly represented. However, there is no standard method for generating structured and well-distributed training datasets.

Iterative IK methods are highly sensitive to the initial joint configuration, and the number of iterations required to converge increases when the initial guess is far from the desired pose. These methods also become unstable or inefficient near singular configurations.

Most existing studies focus on only one class of IK techniques, such as analytical, numerical, or learning-based methods, rather than combining them in a systematic way. Even in hybrid IK approaches, there is limited comparative evaluation of different iterative refinement methods under identical conditions. As a result, it is still unclear which Jacobian-based solver performs best when used after a learning-based IK approximation.

1.3. Novel Contribution of This Work

This research provides two novel contributions to the field of robotic arms:

- Structured Training Methodology: Instead of training the ANFIS with random datasets, this work introduces a structured training methodology based on workspace decomposition. The robot's workspace is divided into discrete cubic regions. Instead of using random data points, cubic polynomial joint-space trajectories are generated between all eight vertices of each cube. This results in 28 unique trajectories (12 edges, 12 face diagonals, four space diagonals) per cube, providing 280 physically consistent data points that mimic the point-to-point movements typically found in industrial applications. This ensures that the ANFIS model is trained on smooth, realistic motion profiles rather than disjointed random poses.
- A Two-Stage Hybrid IK Framework: Recognizing that ANFIS alone often produces approximate solutions for unseen inputs, this work proposes a second-stage refinement process. In the first stage, the ANFIS model provides an approximate initial guess. In the second stage, this approximation is refined through Jacobian-based iterative numerical methods to get more accurate values. In this work three hybrid frameworks are proposed. Three hybrid frameworks were evaluated: Framework-1 (ANFIS + Jacobian Gradient), Framework-2 (ANFIS + Jacobian Pseudoinverse/Newton–Raphson), and Framework-3 (ANFIS + Damped Least Squares).

1.4. Objectives of This Research Articles

The main objectives of this research are:

- To design a structured and repeatable methodology for generating ANFIS training datasets using workspace decomposition into smaller cubic regions.
- To develop a two-stage hybrid inverse kinematics (IK) framework in which the approximate joint configuration predicted by ANFIS is refined by iterative numerical methods to achieve high accuracy.
- To evaluate and compare various hybrid IK using Jacobian Transpose (Gradient), Jacobian Pseudoinverse (Newton–Raphson), and Damped Least Squares as iterative refinement strategies under identical test conditions.

1.5. Scope and Limitations

- Even though these frameworks can be simulated and evaluated on any N-DoF Serial Revolute robotic arm, experimental validation is carried out on the ABB IRB-1200 (5 kg payload, 0.9 m reach) industrial robot.
- While experimental validation was performed within specific cubic regions for the ABB IRB-1200, the methodology is modular. It is intended to be scaled to cover the entire workspace by repeating the cubic decomposition structure.
- The simulations and code are run on a PC with the following configuration: 13th generation Intel i9-13900HX (2.20 Ghz) processor, 16 GB RAM, 8 GB NVIDIA GeForce RTX 4070 graphic card, 64-bit Windows-11 OS.

1.6. Outline of This Paper

This paper is organized as follows:

Section 2 presents the mathematical modeling of the robot, including forward kinematics, Jacobian matrix derivation, and manipulability analysis.

Section 3 describes the inverse kinematics (IK) methods employed in this study, including the proposed hybrid approach.

Section 4 reports the experimental setup, evaluation metrics, and comparative results.

Section 5 concludes the paper and outlines directions for future work.

2. Mathematical Representations of Robot and IK Methods

The ABB IRB-1200-5-0.9 is a compact, high-precision six-degree-of-freedom (6-DoF) articulated industrial robot designed for assembly, material handling, and machine-tending operations. The robot provides a maximum reach of 0.9 m and a payload capacity of 5 kg, making it suitable for both industrial automation and experimental robotics research [32–34]. Figure 1 shows the ABB IRB-1200-5-0.9 robot installed in the laboratory environment, together with its controller and end-effector. The robot consists of a serial chain of six revolute joints that allow full three-dimensional positioning and orientation of the tool center point (TCP). Owing to the nonlinear and coupled nature of this kinematic structure, inverse kinematics for this robot is typically solved using numerical or learning-based approaches rather than closed-form expressions. In this work, the ABB IRB-1200-5-0.9 is used as a benchmark industrial manipulator to validate the proposed two-stage hybrid inverse kinematics framework based on ANFIS and Jacobian-based iterative refinement.



Figure 1. ABB IRB-1200-5-0.9 in the lab.

2.1. Coordinate Frame Assignment

The coordinate frame assignment of the ABB IRB-1200-5-0.9 based on the Denavit–Hartenberg (DH) convention is shown in Figure 2. Frame {0} is attached to the robot base, and Frame {6} is attached to the tool flange. Each joint variable θ_i represents the rotation between two consecutive frames. This systematic frame assignment provides a compact and mathematically consistent description of the robot geometry, which forms the basis for deriving the forward kinematics and Jacobian matrix used in the proposed hybrid IK framework.

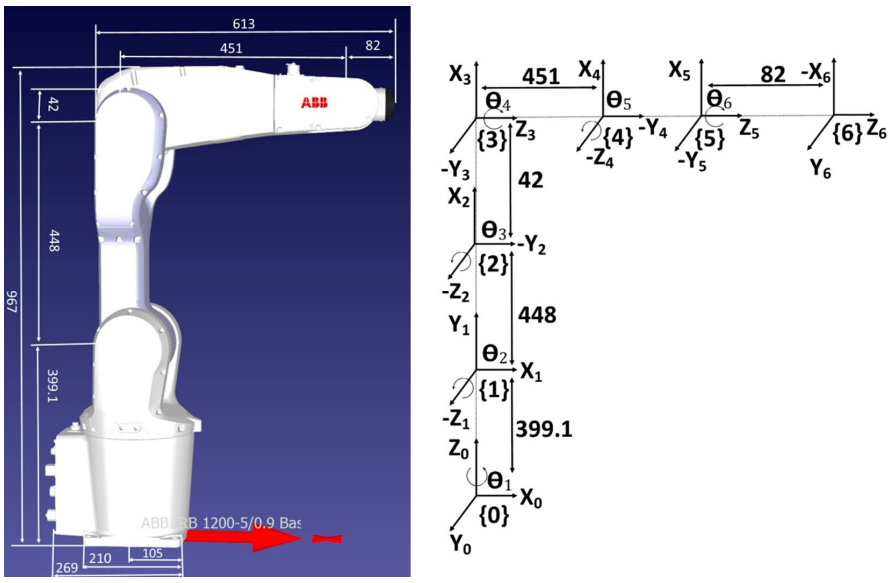


Figure 2. Frame assignment for ABB IRB1200-5-0.9 [32–34].

Using the frame assignment shown in Figure 2, the geometric structure of the ABB IRB-1200-5-0.9 is represented by the Denavit–Hartenberg parameters $(a_i, \alpha_i, d_i, \theta_i)$, which define the link lengths, link twists, joint offsets, and joint angles, respectively, and are listed in Table 1. These parameters completely describe the kinematic chain of the robot and are used to construct the homogeneous transformation matrices required for forward kinematics.

Table 1. DH table for ABB IRB-1200-5-0.9.

Joint	a_i (mm)	α_i (deg)	d_i (mm)	θ_i (deg)	Range (deg)
1	0	−90	399.1	q1	+170 to −170
2	350	0	0	−90	+135 to −100
3	42	−90	0	q3	+70 to −200
4	0	+90	451	q4	+270 to −270
5	0	−90	0	q5	+130 to −130
6	0	0	82	180	+360 to −360

2.2. Forward Kinematics

The forward kinematic model establishes the relationship between the joint variables and the resulting end-effector pose. Once the Denavit–Hartenberg (DH) parameters are defined, the homogeneous transformation matrix of each link is constructed by sequentially transforming the coordinate frame from joint i to joint $i + 1$. Using the standard DH formulation, the transformation matrix is expressed as:

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

where $c\theta_i$ is cos θ_i, $s\theta_i$ is sin θ_i, $c\alpha_i$ is cos α_i, $s\alpha_i$ is sin α_i and i = 1 to 6;

$$ {}^{0}\mathrm{T}_6 = {}^{0}\mathrm{T}_1 \ {}^{1}\mathrm{T}_2 * {}^{2}\mathrm{T}_3 * {}^{3}\mathrm{T}_4 * {}^{4}\mathrm{T}_5 * {}^{5}\mathrm{T}_6 \quad (2) $$

$$ {}^{0}\mathrm{T}_6 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3) $$

Here, $n = [n_x\ n_y\ n_z]^T$, $s = [s_x\ s_y\ s_z]^T$, and $a = [a_x\ a_y\ a_z]^T$ represent the normal, sliding, and approach vectors of the end-effector coordinate frame, while $d = [d_x\ d_y\ d_z]^T$ denotes the Cartesian position.

To obtain the orientation of the ABB IRB 1200 robot, the rotation matrix R is decomposed using the ZYX Euler angle convention, which is consistent with ABB's wrist architecture. Based on this convention, the yaw (z), pitch (y), and roll (x) angles are computed directly, as shown below in Equations (4)–(6). The complete detailed forward kinematic equations are given in Appendix A.

$$ \theta_z = \tan^{-1}\left(\frac{n_y}{n_x}\right) \quad (4) $$

$$ \theta_y = \sin^{-1}(-n_z) \quad (5) $$

$$ \theta_x = \tan^{-1}\left(\frac{s_z}{a_z}\right) \quad (6) $$

2.3. Position and Orientation Error Calculation

To evaluate inverse kinematic solutions, the deviation between the desired and actual end-effector pose is computed.

A. Position error

$$ e_p = \| P_{\mathrm{des}} - P_{\mathrm{act}} \| \quad (7) $$

where

$$P_{\text{des}} = \begin{bmatrix} x_d \; y_d \; z_d \end{bmatrix}^T \tag{8}$$

$$P_{\text{act}} = \begin{bmatrix} x_a \; y_a \; z_a \end{bmatrix}^T \tag{9}$$

B. Orientation Error (ZYX Euler or RPY)

For ZYX rotation sequence used in ABB IRB 1200, the orientation error is computed using

$$R_{\text{err}} = R_{\text{des}} R_{\text{act}}^T \tag{10}$$

The equivalent axis–angle small-angle error vector is

$$e_o = \frac{1}{2} \begin{bmatrix} R_{\text{err}}(3,2) - R_{\text{err}}(2,3) \\ R_{\text{err}}(1,3) - R_{\text{err}}(3,1) \\ R_{\text{err}}(2,1) - R_{\text{err}}(1,2) \end{bmatrix} \tag{11}$$

The final orientation error magnitude is

$$\| \, e_o \, \| = \sqrt{e_{ox}^2 + e_{oy}^2 + e_{oz}^2} \tag{12}$$

2.4. Derivation of Jacobian Matrix

The forward kinematics of the robot arm provides the information about the position and the orientation of the end-effector frame with respect to the base frame. Mathematically, the forward kinematics equation provides the function relating to the Cartesian space (position and orientation) for the given joint space (joint angles). The velocity relationship between the Cartesian space and the joint space is provided by the Jacobian matrix (J). Mathematically, the Jacobian matrix (J) is a linear mapping between the joint velocities ($\dot{X}$) and Cartesian velocities ($\dot{q}$). This Jacobian matrix is expressed as a 6 xn matrix, where, n is number of joint variables. The Jacobian matrix can even be split into two sub Jacobian matrices relating to the translational (JD$\dot{q}$) and angular (JR$\dot{q}$) velocities of the end frame. The JD$\dot{q}$ represents the upper $3 \times n$ matrix of the Jacobian matrix (J), while JR$\dot{q}$ represents the lower $3 \times n$ matrix of the Jacobian matrix (J).

The relationship between the joint velocity and Cartesian velocity is expressed in the form of

$$\dot{\mathrm{X}} = \mathrm{J} \, \dot{q}$$

where $\dot{q}$ is the joint velocity vector

$$\dot{q} = \begin{bmatrix} \dot{q}_1 \; \dot{q}_2 \; \dot{q}_3 \; \cdots \; \dot{q}_{\mathrm{n}} \end{bmatrix}^{\mathrm{T}}$$

$\dot{q}_{\mathrm{i}}$ = velocity of ith joint angle, i = 1 to n.

$\dot{\mathrm{X}}$ is the end-effector velocity vector.

$$\dot{\mathrm{X}} = \begin{bmatrix} \dot{\mathrm{x}}_{\mathrm{n}} \; \dot{\mathrm{y}}_{\mathrm{n}} \; \dot{z}_{\mathrm{n}} \; \omega \mathrm{x}_{\mathrm{n}} \; \omega \mathrm{y}_{\mathrm{n}} \; \omega \mathrm{z}_{\mathrm{n}} \end{bmatrix}^{\mathrm{T}}$$

$\dot{\mathrm{x}}_{\mathrm{n}} \; \dot{\mathrm{y}}_{\mathrm{n}} \; \dot{z}_{\mathrm{n}}$ is the translational velocity of the end-effector frame with respect to x, y and z axes.

$\omega \mathrm{x}_{\mathrm{n}} \; \omega \mathrm{y}_{\mathrm{n}} \; \omega \mathrm{z}_{\mathrm{n}}$ is the angular velocity of the end-effector frame with respect to x, y and z axes.

$$\mathrm{J} = \begin{bmatrix} \mathrm{J}_D \dot{q} \\ \mathrm{J}_R \dot{q} \end{bmatrix} \tag{13}$$

$$J = \begin{bmatrix} {}^0\hat{k}_0 \times {}^0_0d_n & {}^0\hat{k}_1 \times {}^0_1d_n \ldots . & {}^0\hat{k}_{n-1} \times {}^0_{n-1}d_n \\ {}^0\hat{k}_0 & {}^0\hat{k}_1 \quad \ldots . & {}^0\hat{k}_{n-1} \\ \end{bmatrix} \tag{14}$$

The Jacobian matrix of the robotic arm can also be represented as column by column using the Jacobian generating vector $c_i(q)$.

$$ci(q) = \begin{bmatrix} {}^0\hat{k}_{i-1} \times {}^0_{i-1}d_n \\ {}^0\hat{k}_{i-1} \end{bmatrix} \tag{15}$$

where i = 1 to n, and n = 6;

${}^0_{i-1}d_n$ are the positional vectors of i-1th frame from the base frame;

${}^0\hat{k}_{i-1}$ is the joint axis unit vector attached to the frame $(i-1)$.

The final Jacobian matrix for the 6-DOF robotic arm is calculated using Equation (15), while each column of the Jacobian is shown in Equation (16). The detailed Jacobian matrix is listed in Appendix B.

$$J = \begin{bmatrix} j_{11} & j_{12} & j_{13} & j_{14} & j_{15} & j_{16} \\ j_{21} & j_{22} & j_{23} & j_{24} & j_{25} & j_{26} \\ j_{31} & j_{32} & j_{32} & j_{34} & j_{35} & j_{36} \\ j_{41} & j_{42} & j_{43} & j_{44} & j_{45} & j_{46} \\ j_{51} & j_{52} & j_{53} & j_{54} & j_{55} & j_{56} \\ j_{61} & j_{62} & j_{63} & j_{64} & j_{65} & j_{66} \end{bmatrix} \tag{16}$$

2.5. Manipulability Factor

Manipulability describes how easily the robot can produce motion in any direction at a given configuration. The Yoshikawa manipulability index is used:

$$w(q) = \sqrt{\det(J(q) * J(q)^T)}. \tag{17}$$

where J(q) is the Jacobian Matrix

A low $w(q)$ indicates that the robot is close to a singularity, where small joint motions lead to limited Cartesian motion. This metric is used in:

Workspace classification;Singularity avoidance;

Hybrid IK switching logic;

Evaluation of joint-space trajectories.

3. Existing IK and Proposed Hybrid IK Methods and Experiment

This section gives an overview of the existing IK methods and proposed hybrid IK methods followed by the experiments that were conducted for this research work. We have also proposed a novel method to generate training dataset for ANFIS IK techniques.

3.1. Existing IK Methods

In the existing IK section, first an overview of a soft-computing-based IK method is discussed followed by iterative methods like the Gradient method based on Jacobian Transpose, the Newton–Raphson method based on pseudoinverse of the Jacobian, and the Damped Least Squares Method.

3.1.1. ANFIS IK Method

The Adaptive Neuro-Fuzzy Inference System (ANFIS) is employed in this work to learn the nonlinear inverse mapping between the Cartesian pose of the end-effector and the corresponding joint configuration of the robot. For a 6-DoF manipulator, the inverse

kinematics problem consists of estimating the joint vector $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ from the Cartesian pose $(x, y, z, \theta x, \theta y, \theta z)$

Instead of using a single monolithic network, this work proposes a joint-wise ANFIS architecture, illustrated in Figure 3, in which six independent ANFIS models are trained—one for each joint. This decomposition reduces the dimensionality of each learning problem and improves training stability and accuracy. For each joint θ_i, a dedicated training database is constructed from forward kinematics data, pairing the Cartesian pose x with the corresponding joint value θ_i. Each ANFIS therefore learns a nonlinear mapping

$$\theta_i = f_i(x), i = 1, \ldots, 6.$$

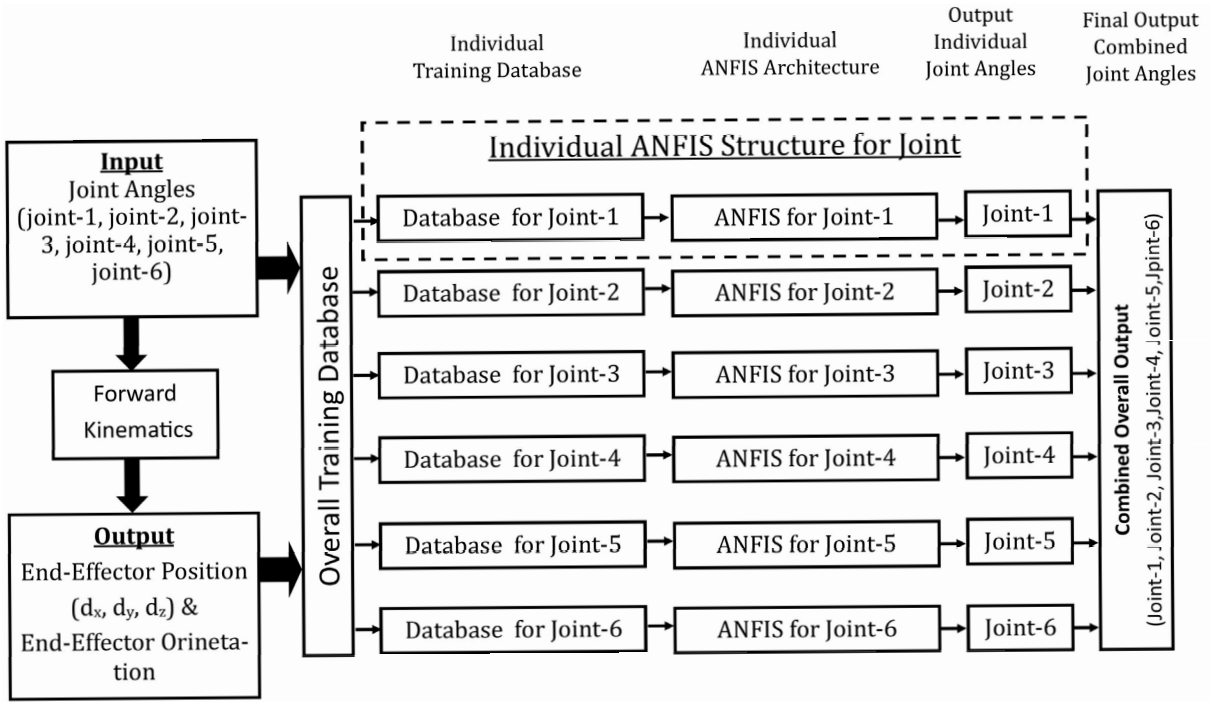


Figure 3. ANFIS IK structure.

For a desired End-effector pose x_{des}, the six ANFIS models operate in parallel to generate the estimated joint vector

$$\hat{q}_{\text{ANFIS}} = \left[\hat{\theta}_1, \hat{\theta}_2, \hat{\theta}_3, \hat{\theta}_4, \hat{\theta}_5, \hat{\theta}_6\right]^T.$$

Each ANFIS follows a standard five-layer neuro-fuzzy structure consisting of fuzzy membership functions, rule firing layers, normalization, and linear consequent functions. The parameters are trained using a hybrid learning algorithm that combines least-squares estimation for consequent parameters with gradient descent for membership functions.

ANFIS provides extremely fast inference and captures the nonlinear inverse kinematic relationship of the robot. However, its accuracy depends strongly on the coverage and structure of the training dataset, and predictions outside the trained workspace are only approximate. Therefore, in the proposed two-stage hybrid framework, the ANFIS output is used as an initial joint estimate, which is subsequently refined using Jacobian-based numerical solvers.

3.1.2. Jacobian Gradient Method Based on Transpose of Jacobian

The Gradient Jacobian method computes inverse kinematics by iteratively reducing the end-effector pose error using the direction of the steepest descent. For a desired end-

effector position x_{des} and the current forward-kinematics output x_{cur}, the position error is determined.

The manipulator Jacobian $J(q)$ maps joint velocities to Cartesian velocities. By using the Jacobian Transpose, the joint update direction aligns with the negative gradient of the error function. The incremental joint correction is computed as

$$\Delta q = \alpha J^T(q)e \tag{18}$$

where $\alpha > 0$ is a step-size gain chosen to ensure stability and avoid overshoot.

The new joint vector is obtained by

$$q_{new} = q_{old} + \Delta q. \tag{19}$$

This iterative update continues until the pose error falls below a chosen tolerance. Because the method avoids Jacobian inversion and relies only on transpose operations, it is computationally efficient and suitable for real-time control. However, convergence may slow near singularities or when the initial guess is far from the solution, making it ideal for hybrid IK frameworks where ANFIS provides a closer starting point. Pseudocode of the architecture is listed for detailed understanding.

Pseudocode:

1. Initialize q
2. Repeat until error < tolerance:
    a. Compute FK(q) $\rightarrow$ x_cur
    b. Compute error e = x_des $-$ x_cur
    c. Compute Jacobian J(q)
    d. $\Delta q = \alpha * J^T * e$
    e. $q = q + \Delta q$
3. Return q_new

3.1.3. Newton–Raphson Method Based on Pseudo Inverse of Jacobian

The Jacobian Pseudoinverse method is a widely used numerical IK technique that provides fast and stable convergence by directly minimizing the end-effector pose error.

The joint correction is obtained through the Moore–Penrose pseudoinverse of the Jacobian

$$\Delta q = J^{\dagger}e = (J^TJ)^{-1}J^Te. \tag{20}$$

The new joint vector is therefore updated as

$$q_{new} = q_{old} + \Delta q. \tag{21}$$

Unlike the Jacobian Transpose method, the pseudoinverse uses complete Jacobian information, resulting in faster convergence and better accuracy. It is effective for well-conditioned configurations but becomes unstable near singularities because $(J^TJ)^{-1}$ grows unbounded. This limitation motivates the need for damping or hybrid refinements in complex manipulators such as ABB IRB-1200.

This method converges faster than the Gradient method and provides accurate results but becomes unstable near singularities due to the inversion of J^TJ. It is suitable when the robot operates in well-conditioned regions of the workspace. Pseudocode of the architecture is listed for detailed understanding.

Pseudocode:

1. Initialize q
2. Repeat until error < tolerance:
    a. Compute FK(q) → x_cur
    b. Compute error e
    c. Compute Jacobian J
    d. Compute pseudoinverse $J^+ = (J^T J)^{-1} J^T$
    e. $\Delta q = J^+ * e$
    f. $q = q + \Delta q$
3. Return q_new

3.1.4. Damped Least Squares Method

The Damped Least Squares approach enhances stability in numerical IK, especially near singular configurations. It modifies the classical pseudoinverse formula by introducing a damping term $\lambda^2 I$, producing a more robust inversion. The joint update is obtained as

$$\Delta q = (J^T J + \lambda^2 I)^{-1} J^T e, \tag{22}$$

where λ is a small positive damping factor.

The updated joint vector is computed as

$$q_{new} = q_{old} + \Delta q. \tag{23}$$

The damping term prevents numerical blow-up and guarantees smooth convergence even when the robot is close to singularities or when the Jacobian becomes ill-conditioned. Although DLS slightly slows convergence compared to the pseudoinverse, it significantly improves stability, making it highly suitable for real robots and hybrid IK systems where approximation errors may occur in initial guesses. Pseudocode of the architecture is listed for detailed understanding.

Pseudocode:

1. Initialize q
2. Define damping λ
3. Repeat until error < tolerance:
    a. Compute FK(q)
    b. Compute error e
    c. Compute Jacobian J
    d. Compute DLS inverse = $(J^T J + \lambda^2 I)^{-1} J^T$
    e. Δq = DLS_inverse * e
    f. $q = q + \Delta q$
4. Return q_new

3.2. Proposed Hybrid IK Methods

This section describes in detail the proposed IK architecture. First we have proposed a novel cube-based structured training strategy for the ANFIS IK method, followed by three proposed hybrid IK methods.

3.2.1. ANFIS IK Method with Structured Training Database

Most learning-based inverse kinematics (IK) methods rely on randomly sampled training data, which leads to uneven workspace coverage and poor generalization in untrained

regions. In contrast, this work adopts a structured workspace decomposition strategy, where the robot workspace is divided into small cubic regions. Each cube represents a local learning domain in which the Cartesian joint mapping remains smooth and well-conditioned. A conceptual illustration of such workspace partitioning is shown in Figure 4a (Gemini-generated visualization of cubic workspace decomposition).

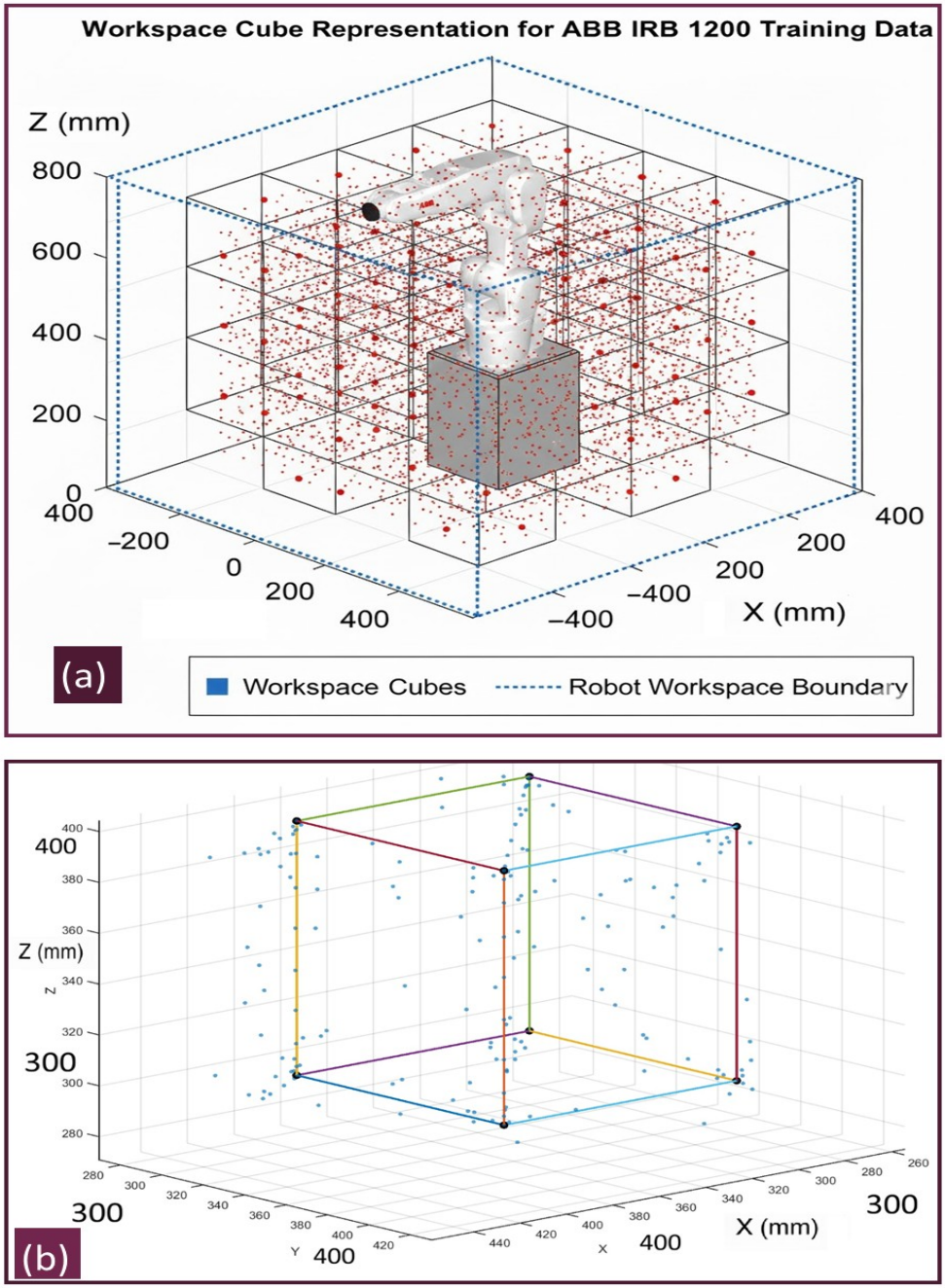


Figure 4. (**a**) Gemini AI generated image for visualization of ABB IRB1200 Robot using cube-based decomposition for generating training dataset. (**b**) *Cubic workspace with the training dataset used in this work.*

In this study, one representative cube is selected to demonstrate the methodology, as shown in Figure 4b. The cube spans the Cartesian limits

$$(300, 300, 300) \text{ mm to } (400, 400, 400) \text{ mm.}$$

This region lies well within the reachable workspace of the ABB IRB-1200 and avoids singular configurations, making it suitable for controlled learning and validation. The proposed methodology can be directly extended to cover the entire workspace by repeating the same procedure for all cubes.

Step 1: Determination of Cube Vertex Poses

Each cube has eight vertices, corresponding to the extreme combinations of its Cartesian limits. The end-effector poses (position and orientation) at these eight vertices were first obtained using RoboDK simulation, and then verified on the real ABB IRB-1200 robot. To further ensure consistency, the corresponding joint angles were also validated using a Damped Least Squares (DLS) inverse kinematics solver, which required approximately 6 ms per pose due to the need for iterative Jacobian inversion.

These eight vertices' poses and joint angles form the anchor points of the cube and define its kinematic boundary. These vertex points are listed in the Table 2.

Table 2. Cartesian and joint-space coordinates of the eight vertex points of the cube.

Vertex	dx	dy	dz	rx	ry	rz	j1	j2	j3	j4	j5	j6
p1	300	300	300	90	90	90	54.00	39.49	44.89	−125.87	86.70	175.45
p2	300	400	300	135	0	180	48.74	35.08	34.39	−61.81	−31.94	115.92
p3	400	400	300	45	45	45	51.31	51.75	20.70	54.00	−110.50	18.90
p4	400	300	300	200	45	45	34.81	42.59	18.46	193.03	−76.37	−10.49
p5	300	300	400	90	90	90	54.00	23.45	46.71	55.65	−78.49	−16.27
p6	300	400	400	135	0	180	48.74	22.69	34.10	97.03	28.02	−39.76
p7	400	400	400	45	45	45	51.31	39.88	23.35	51.65	−104.93	11.20
p8	400	300	400	200	45	45	34.81	32.44	18.30	192.68	−86.43	−8.17

Step 2: Cubic Polynomial Joint-Space Trajectory Generation

To generate dense and physically consistent training data inside the cube, joint-space trajectories are generated between all vertex pairs using cubic polynomial interpolation.

For each joint $q_i(t)$, the trajectory between an initial configuration $q_i(t_0) = q_{i0}$ and a final configuration $q_i(t_f) = q_{if}$ with zero endpoint velocities is defined as

$$q_i(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3, t \in [0,\ 1] \tag{24}$$

with boundary conditions

$$q_i(0) = q_{i0}, q_i(1) = q_{if}, \dot{q}_i(0) = 0, \dot{q}_i(1) = 0. \tag{25}$$

Solving these conditions yields

$$q_i(t) = q_{i0} + 3(q_{if} - q_{i0})t^2 - 2(q_{if} - q_{i0})t^3. \tag{26}$$

This guarantees smooth motion with continuous velocity and acceleration profiles.

Step 3: Structured Training Dataset Construction

A cube has 8 vertices, producing 28 unique vertex pairs (12 edges, 12 face diagonals, 4 space diagonals) listed in Table 3. For each pair, a cubic joint-space trajectory is generated and uniformly sampled at 10 time instants. This produces smooth Cartesian paths that pass not only along cube edges but also through the interior of the cube, ensuring that the ANFIS model is trained on representative internal points rather than only boundary points. For

each sampled joint vector, forward kinematics is used to compute the corresponding Cartesian pose. The resulting dataset therefore consists of pairs that are physically consistent, continuous, and uniformly distributed within or near the cube.

Table 3. Twenty-eight unique vertex pair trajectories used for generating training dataset for ANFIS.

Trajectory	Count	Vertex Pairs (Trajectories)	Path Description
Edges	12	(P1, P2), (P2, P3), (P3, P4), (P4, P1) (P5, P6), (P6, P7), (P7, P8), (P8, P5) (P1, P5), (P2, P6), (P3, P7), (P4, P8)	Boundary paths defining the outer frame of the cube.
Face diagonals	12	(P1, P3), (P2, P4), (P5, P7), (P6, P8) (P1, P6), (P2, P5), (P3, P8), (P4, P7) (P1, P8), (P4, P5), (P2, P7), (P3, P6)	Cross-planar paths covering the six exterior surfaces.
Space diagonals	4	(P1, P7), (P2, P8), (P3, P5), (P4, P6)	Interior paths passing through the cube's center.
Total	28		Comprehensive volumetric coverage

Step 4: ANFIS Training and IK Approximation

Six independent ANFIS models are trained for each joint using the structured dataset. Each ANFIS learns a local inverse mapping from a Cartesian pose to its corresponding joint angle. The architecture is shown in Figure 5, where each joint has its own ANFIS trained using the same structured cube dataset. Because the cube is small and the data are smooth and well-distributed, ANFIS produces highly accurate local inverse kinematics estimates inside the cube. However, ANFIS alone remains an approximation, which motivates the second-stage Jacobian-based refinement in the proposed hybrid frameworks.

3.2.2. Hybrid IK Framework-1: ANFIS + Jacobian Gradient Method

In this hybrid framework, in Stage 1, the ANFIS model is used to provide an approximate inverse kinematics solution, which is subsequently refined using the Jacobian Gradient (Jacobian Transpose) method in Stage 2. The role of ANFIS is to move the solution close to the true inverse kinematic configuration, while the Gradient method performs local error minimization. Figure 6 shows the flow chart for the hybrid IK architecture. Because the Jacobian Transpose avoids matrix inversion, it is computationally lightweight and numerically stable. However, it converges slowly when initialized far from the solution. In the proposed hybrid framework, ANFIS ensures that the initial joint vector lies close to the correct solution, allowing the Gradient method to act as a fast local corrector.

3.2.3. Hybrid IK Framework-2: ANFIS + Newton–Raphson (Jacobian Pseudoinverse)

In this framework, the ANFIS model again supplies an initial joint estimate, which is refined using the Newton–Raphson method based on the Jacobian Pseudoinverse. Figure 7 shows the flowchart for this hybrid architecture. This approach exploits the fast convergence properties of Newton's method while avoiding the need for a global numerical search. This method achieves very fast convergence when the Jacobian is well-conditioned. However, if the robot approaches a singular configuration, the term $(J^T J)^{-1}$ becomes ill-conditioned, causing numerical instability. The hybrid formulation overcomes this limitation by ensuring that ANFIS places the robot close to the correct solution region, where the Jacobian is well-conditioned. Consequently, Newton–Raphson convergence becomes both fast and reliable.

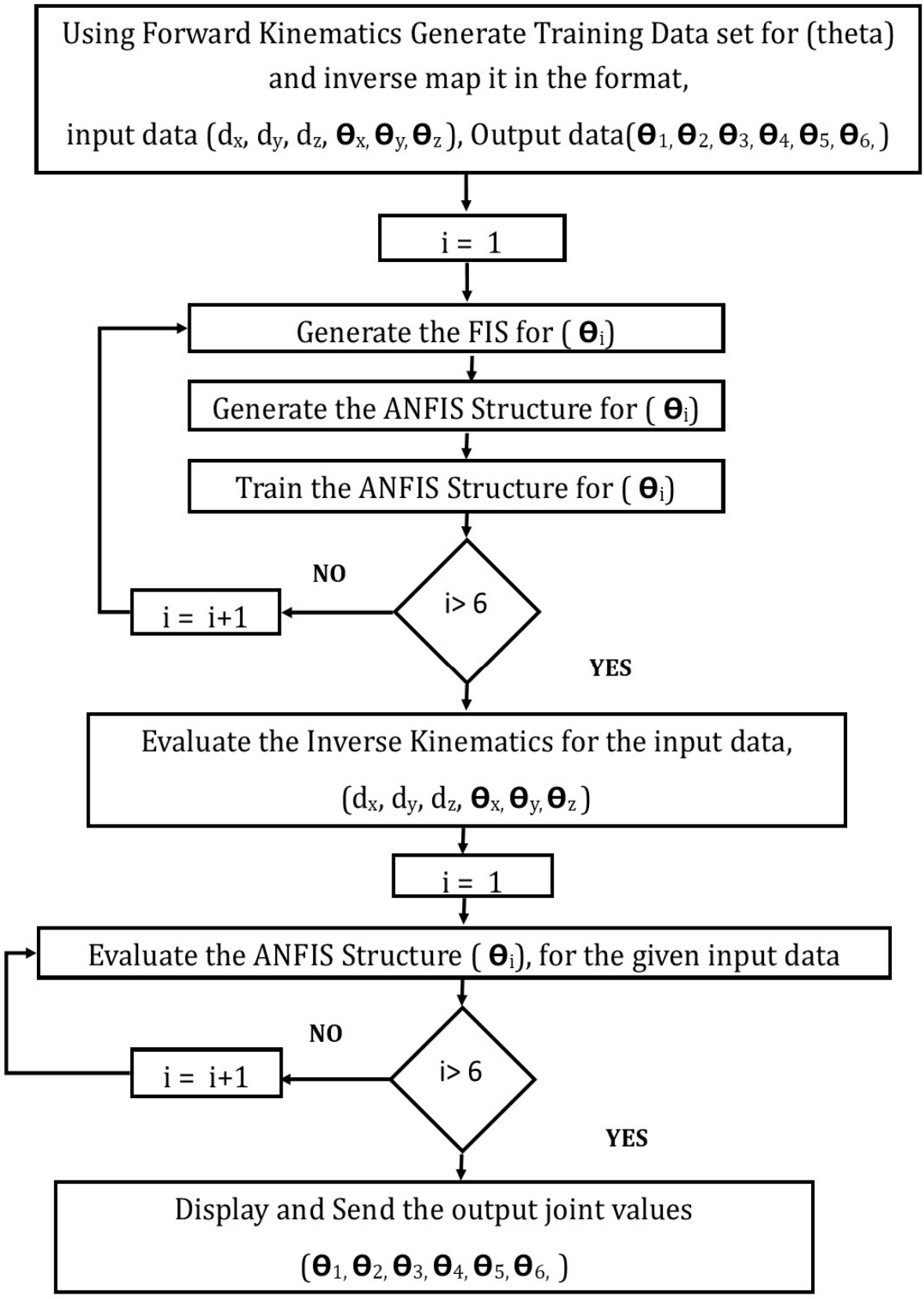


Figure 5. Flowchart of ANFIS IK architecture.

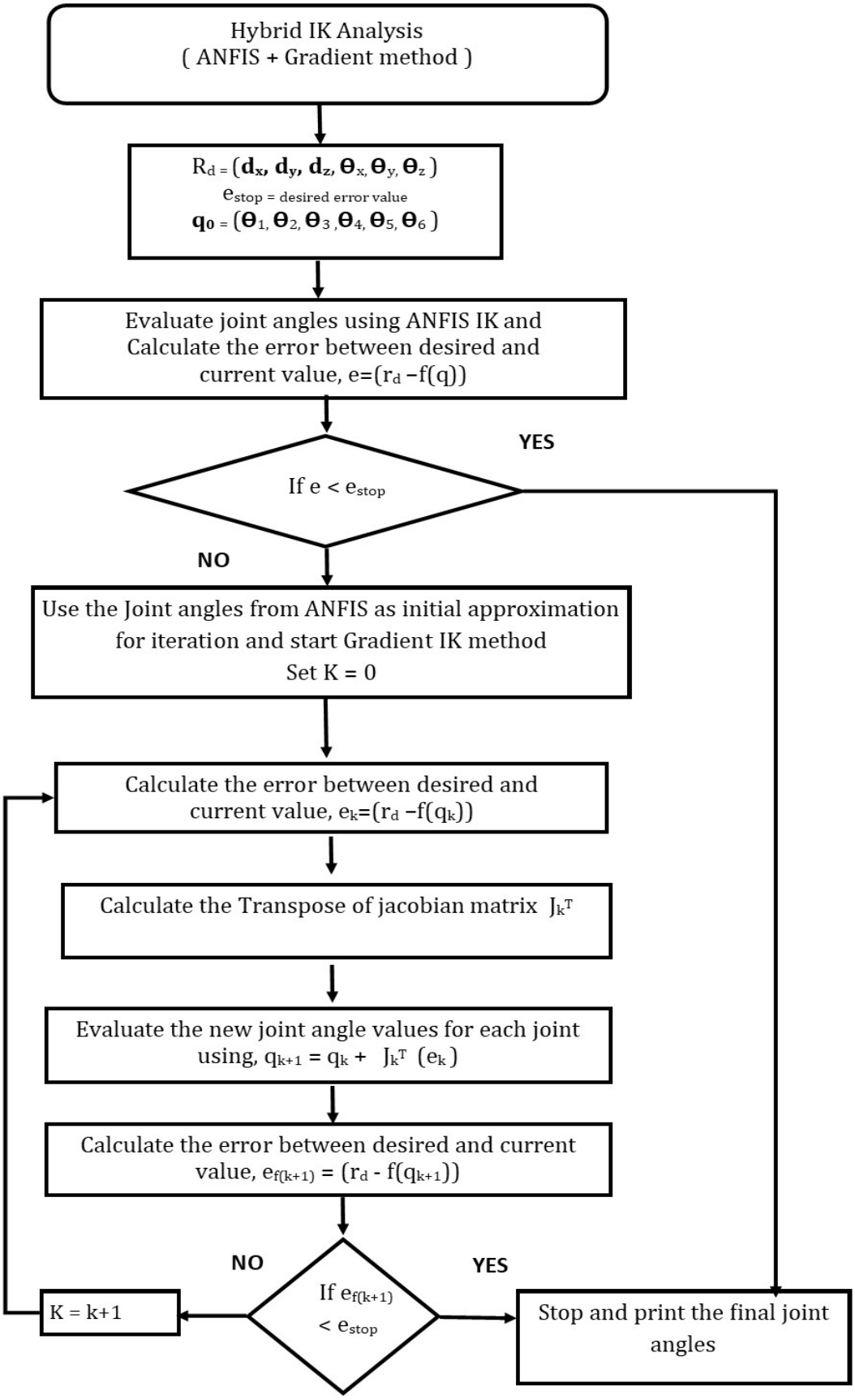


Figure 6. Flowchart of hybrid IK framework based on ANFIS + Gradient methods.

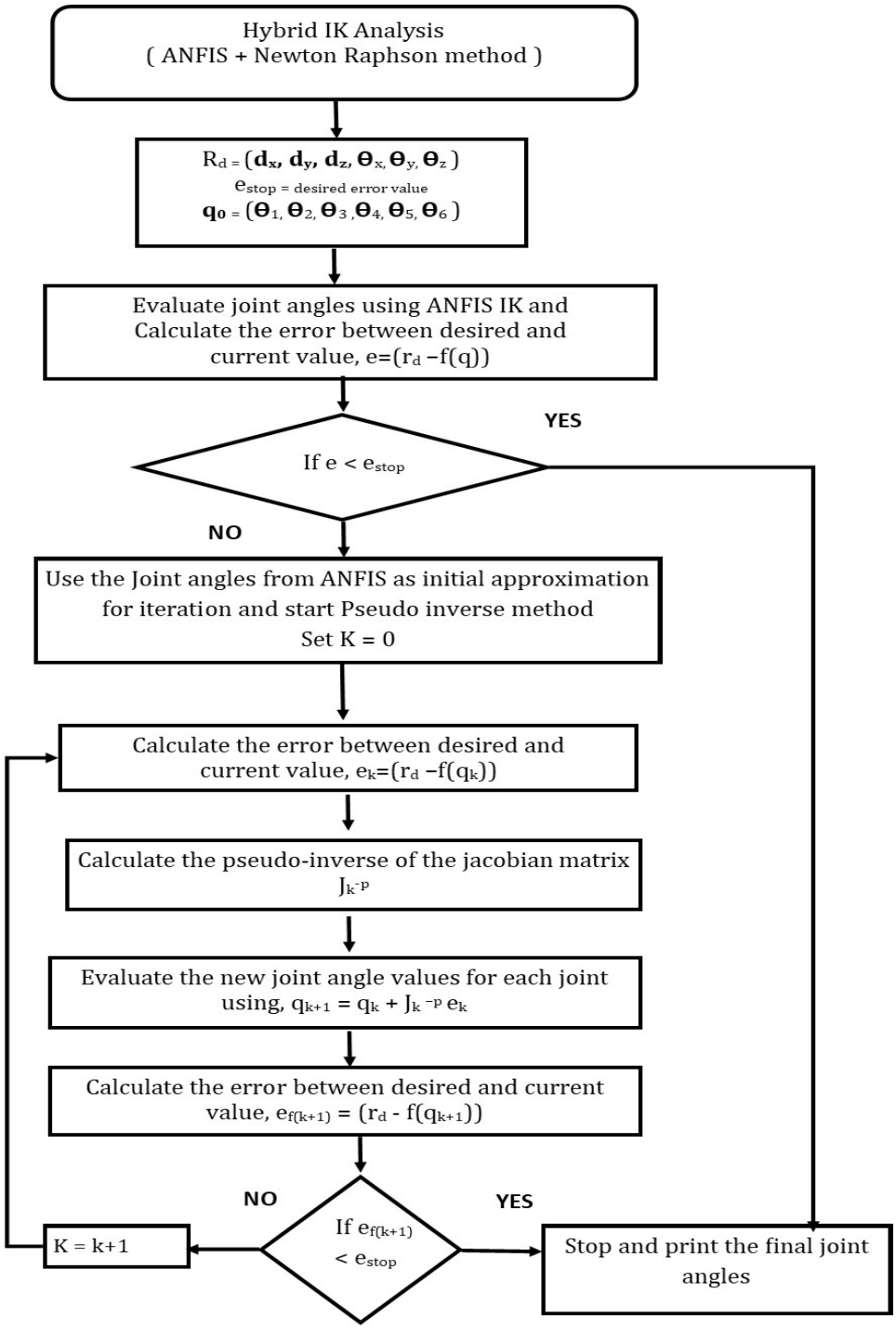


Figure 7. Flowchart of hybrid IK framework (ANFIS + Newton–Raphson).

3.2.4. Hybrid IK Framework-3: ANFIS + Damped Least Squares (DLS)

The third hybrid framework combines ANFIS with the Damped Least Squares (DLS) inverse kinematics method, as shown in Figure 8. This configuration provides the highest numerical robustness, particularly near singularities and in regions of low manipulability. The damping term prevents numerical instability by limiting the effect of small singular values in the Jacobian. Although this slightly reduces convergence speed compared to the Newton–Raphson method, it guarantees smooth and stable behavior even near singular configurations. When coupled with ANFIS-based initialization, the DLS method achieves rapid convergence, low pose error, and excellent numerical stability, making it particularly suitable for real-world industrial manipulators such as the ABB IRB-1200.

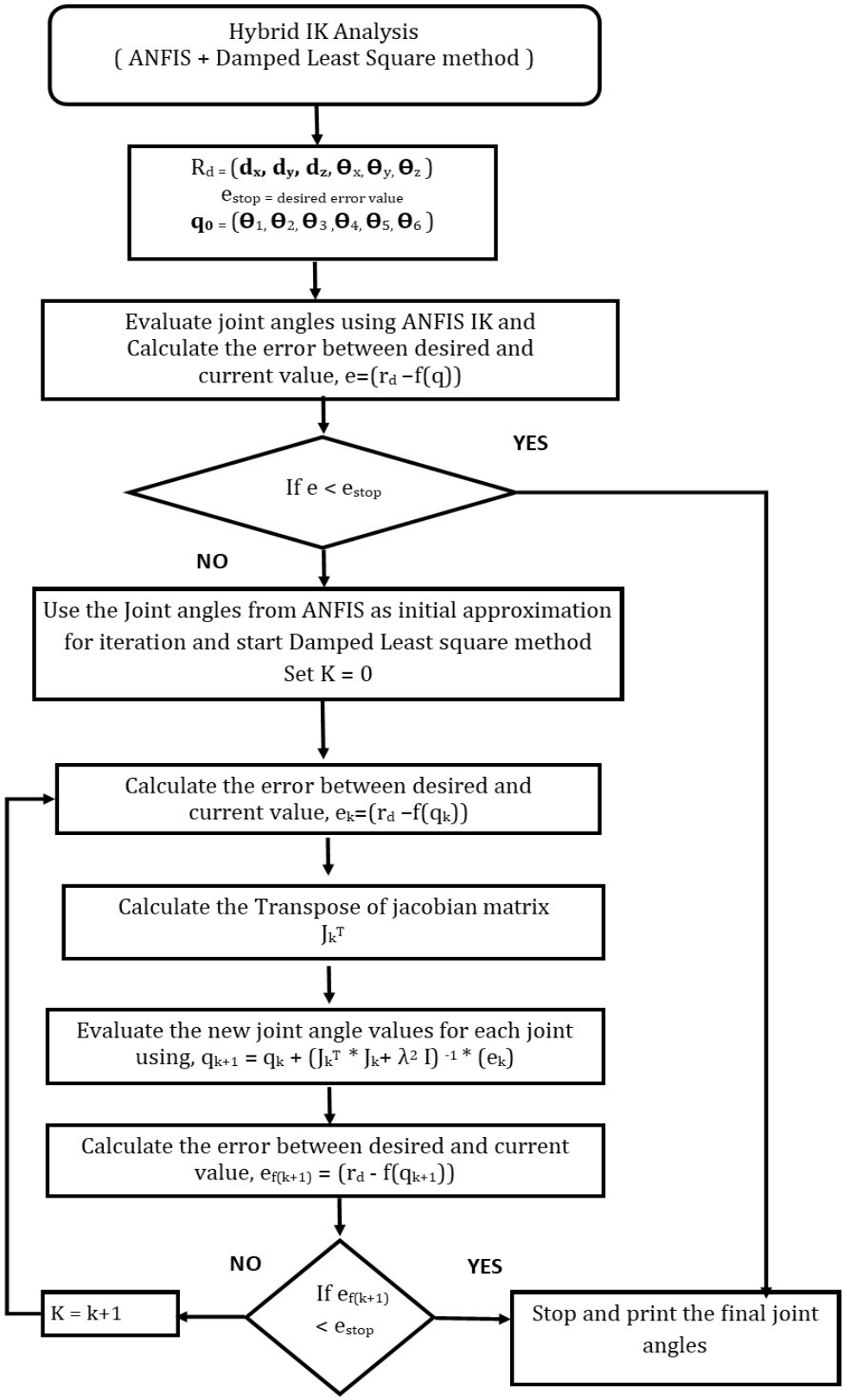


Figure 8. Flowchart for hybrid IK based on (ANFIS and Damped Least Squares Method).

4. Experiment, Results and Discussion

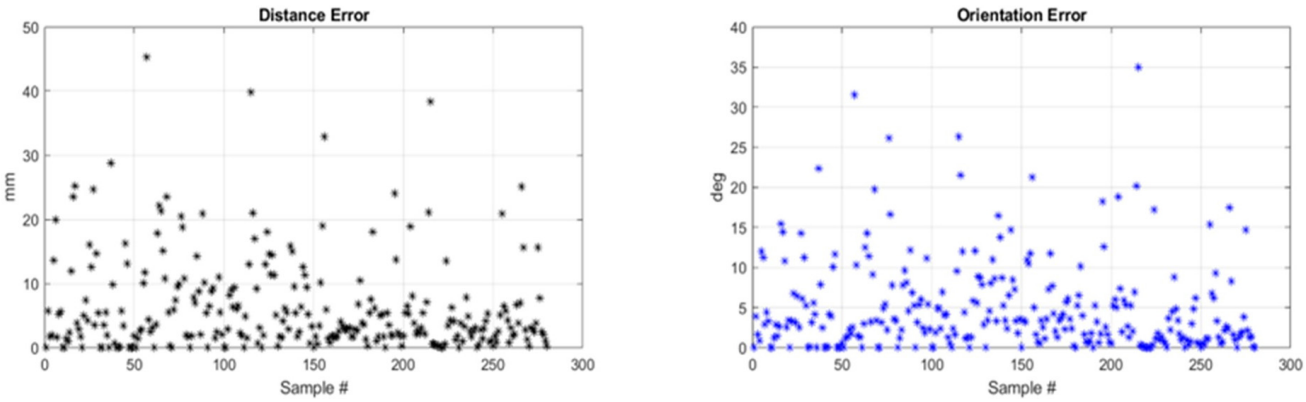
To validate the proposed two-stage hybrid IK framework, a sequence of experiments was performed using the ABB IRB-1200 robot within the cubic workspace region of

(300, 300, 300) mm to (400, 400, 400) mm. The experiments were designed to compare ANFIS-based IK and conventional Jacobian-based IK with its hybrid architecture.

4.1. Performance of ANFIS as a Stand-Alone IK Solver

In Experiment 1, the trained ANFIS model was evaluated on the cube-based training samples to obtain joint angles, forward kinematics, and pose error.

The results from Experiment 1 are shown in Figure 9, which presents the Cartesian pose error obtained when ANFIS alone is used to predict joint angles for the given target poses. As shown in Figure 9a, the distance error varies from near zero to approximately 40–50 mm, while Figure 9b indicates that orientation errors reach values of approximately 30–35°. These results demonstrate that although ANFIS successfully maps Cartesian poses to reasonable joint configurations, its standalone accuracy is insufficient for high-precision industrial manipulation.



(**a**) (**b**)

Figure 9. IK results obtained using ANFIS method. (**a**) Distance Error, (**b**) Orientation Error.

Nevertheless, the primary role of ANFIS in the proposed framework is not to provide final accuracy but rather to localize the solution. As observed in Figure 9, the ANFIS output consistently places the joint configuration within a narrow neighborhood of the true IK solution. This coarse but reliable approximation significantly reduces the search space of the subsequent Jacobian-based refinement stage, which forms the fundamental principle of the proposed two-stage hybrid architecture.

4.2. Jacobian Gradient Method

In Experiment 2, the Jacobian Gradient (Transpose) method was applied to the same samples using the robot home configuration as the initial joint vector. The results obtained from Experiment 2 are shown in Figure 10. When the Jacobian Gradient solver is initialized from the home configuration, it can be observed that the positional error is reduced to sub-millimeter levels for most samples. However, the orientation error remains high, with most values lying between 40° and 60°. The iteration plot further reveals that all samples reach the maximum iteration limit of 50, indicating stagnation of the solver. This behavior suggests that the gradient-based method becomes trapped in a local minimum when initialized far from the desired pose, particularly in the orientation subspace.

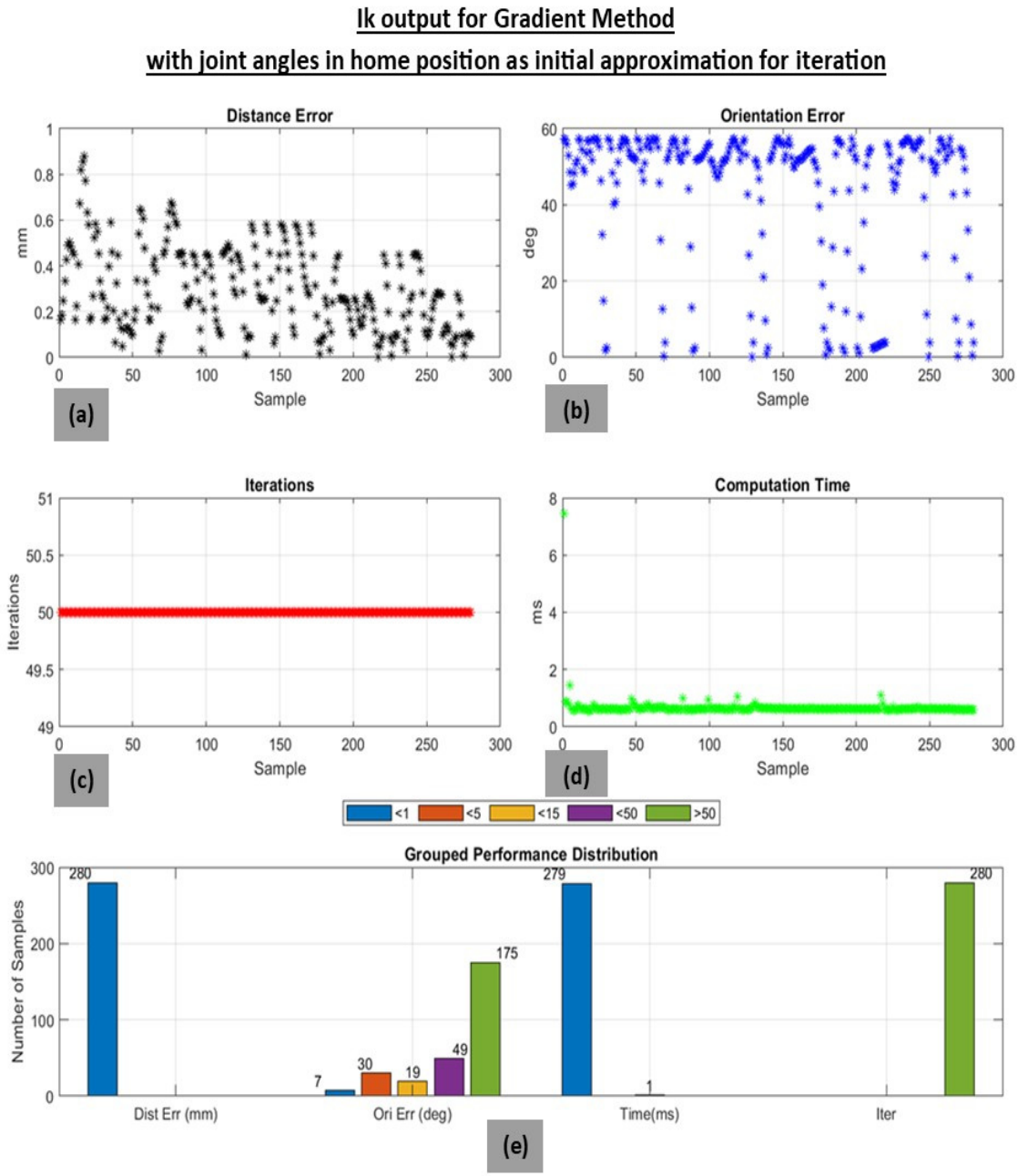


Figure 10. IK results obtained for Gradient method. (**a**) Distance Error, (**b**) Orientation Error, (**c**) No of iterations required, (**d**) Computation Time required, (**e**) Grouped performance distribution.

4.3. Hybrid Framework-1 (ANFIS + Jacobian Gradient)

In Experiment 3, the Jacobian Gradient method was again applied, but the initial joint vector was taken from the ANFIS output from Experiment 1, forming the Hybrid ANFIS + Gradient IK. The results obtained from Experiment 3, in which ANFIS is used as initial approximation and the Jacobian Gradient method is used, are shown in Figure 11. When initialized using ANFIS-predicted joint values, the performance of the gradient solver improves substantially. The distance error decreases to below 0.05 mm for most samples, and orientation errors are largely reduced to below 15°. The iteration count is dramatically reduced from 50 to typically 1–3 iterations, indicating rapid convergence. These results confirm that ANFIS converts the Gradient method from a poorly convergent global optimizer into an efficient local refinement solver.

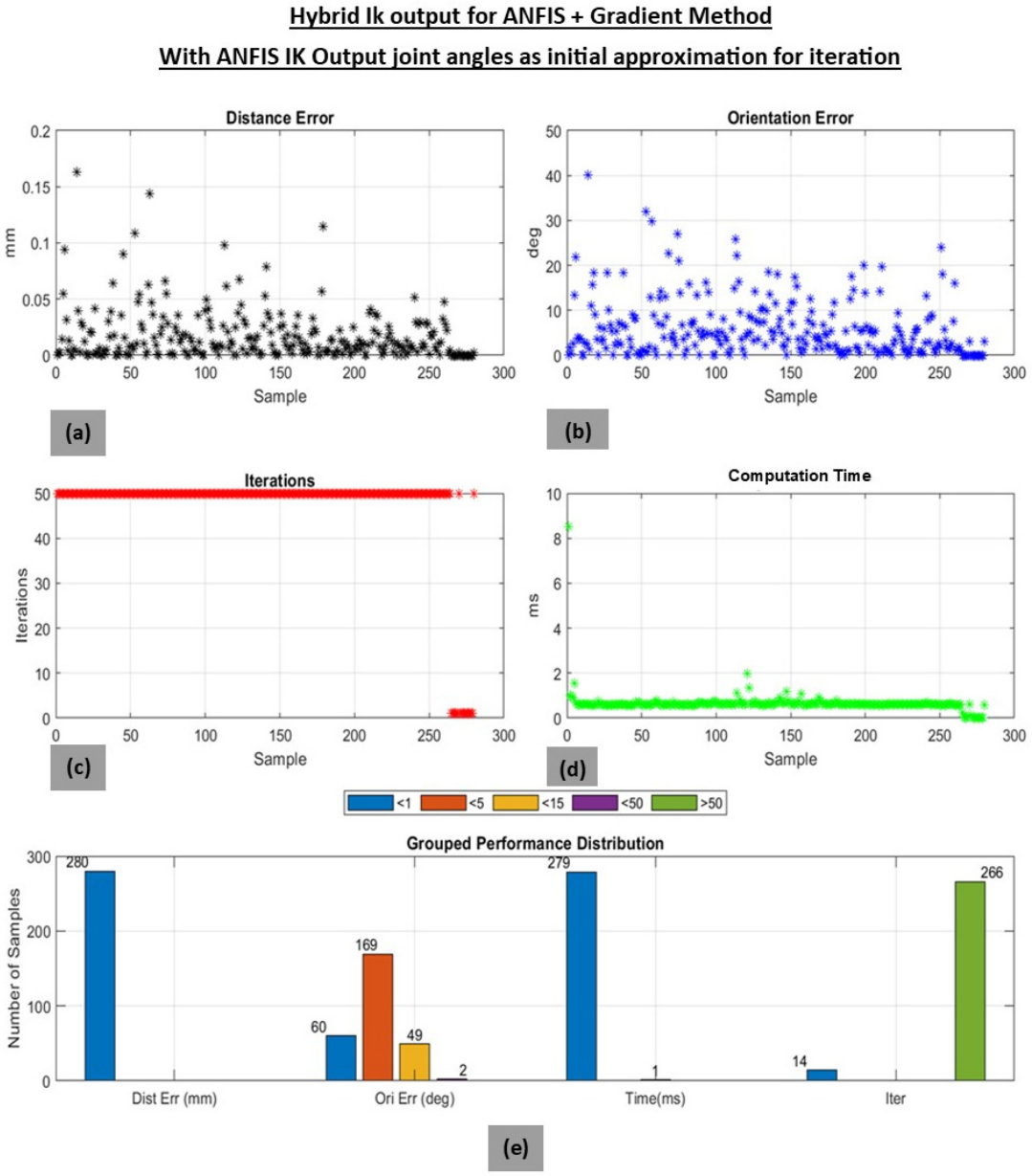


Figure 11. Hybrid IK (ANFIS + Gradient) results. (**a**) Distance Error, (**b**) Orientation Error, (**c**) No of iterations required, (**d**) Computation Time required, (**e**) Grouped performance distribution.

4.4. Jacobian Pseudoinverse (Newton–Raphson) Method

In Experiment 4, the Newton–Raphson method based on the Jacobian Pseudoinverse was applied using the home configuration as the initial joint vector. The Jacobian Pseudoinverse method is evaluated to analyze the behavior of a second-order Newton-based solver within the proposed two-stage hybrid IK framework. Results obtained from Experiment 4 are shown in Figure 12. When the Newton–Raphson solver is initialized from the home position, severe numerical instability occurs, as shown in Figure 12a. The distance error reaches several hundred millimeters, and the orientation error becomes widely scattered. The iteration count saturates at the maximum limit of 1000, and the computation time increases to tens of milliseconds. This behavior is attributed to the inversion of an ill-conditioned Jacobian when the initial configuration is far from the true solution.

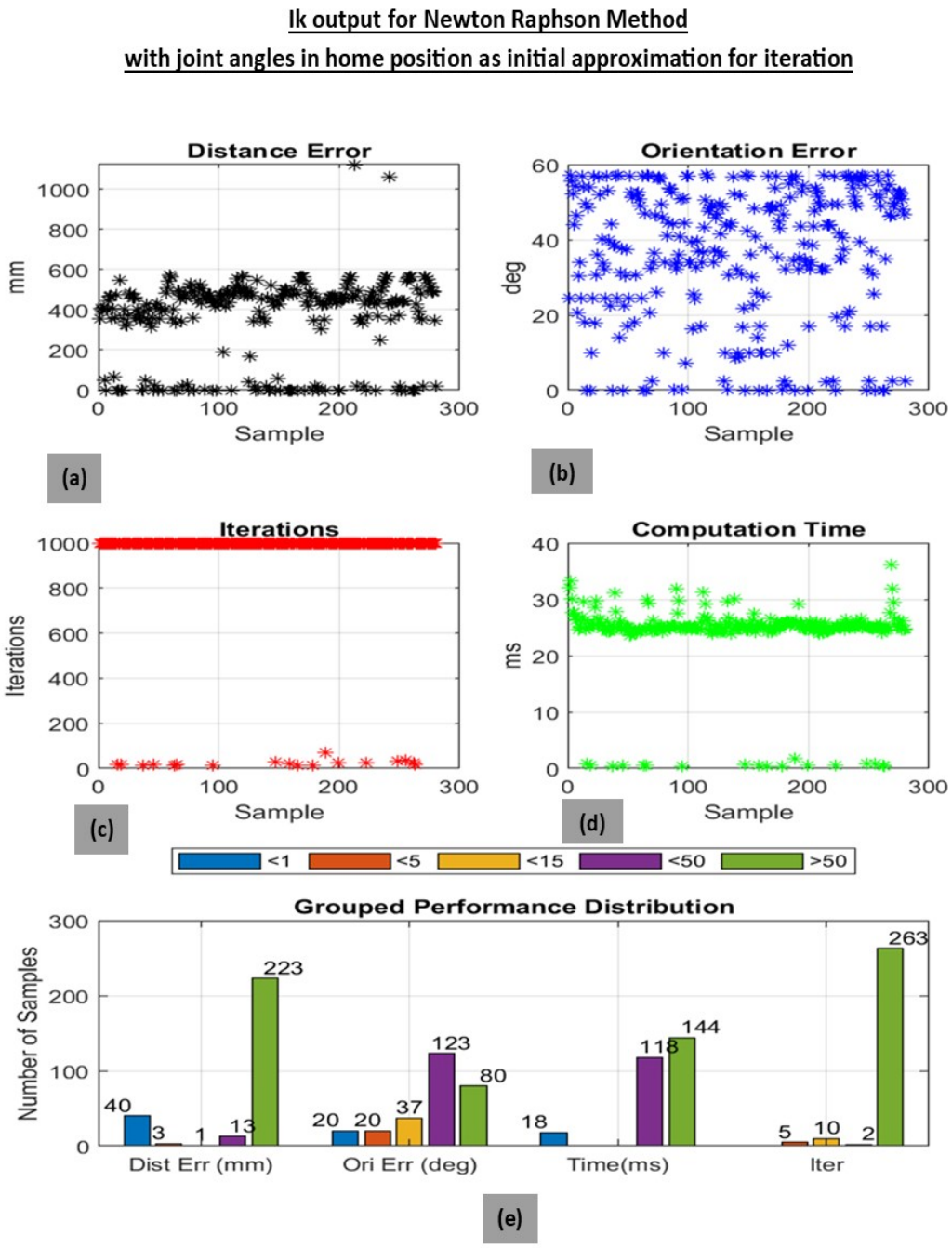


Figure 12. IK results obtained for Newton–Raphson method. (**a**) Distance Error, (**b**) Orientation Error, (**c**) No of iterations required, (**d**) Computation Time required, (**e**) Grouped performance distribution.

4.5. Hybrid Framework-2 (ANFIS + Newton–Raphson Using Jacobian Psuedoinverse)

In Experiment 5, the pseudoinverse method was initialized using the ANFIS output from Experiment 2, forming the Hybrid ANFIS + Pseudoinverse IK. Results obtained from Experiment 5 are shown in Figure 13. In contrast, when ANFIS is used to provide the initial joint estimate, the pseudoinverse method becomes both highly accurate and computationally efficient. As shown in Figure 13b, both distance and orientation errors are nearly zero for almost all samples. The iteration count typically falls below five, and computation time is reduced to less than 1 ms. This confirms that ANFIS places the solver in a well-conditioned region of the solution space, enabling Newton's method to exploit its quadratic convergence.

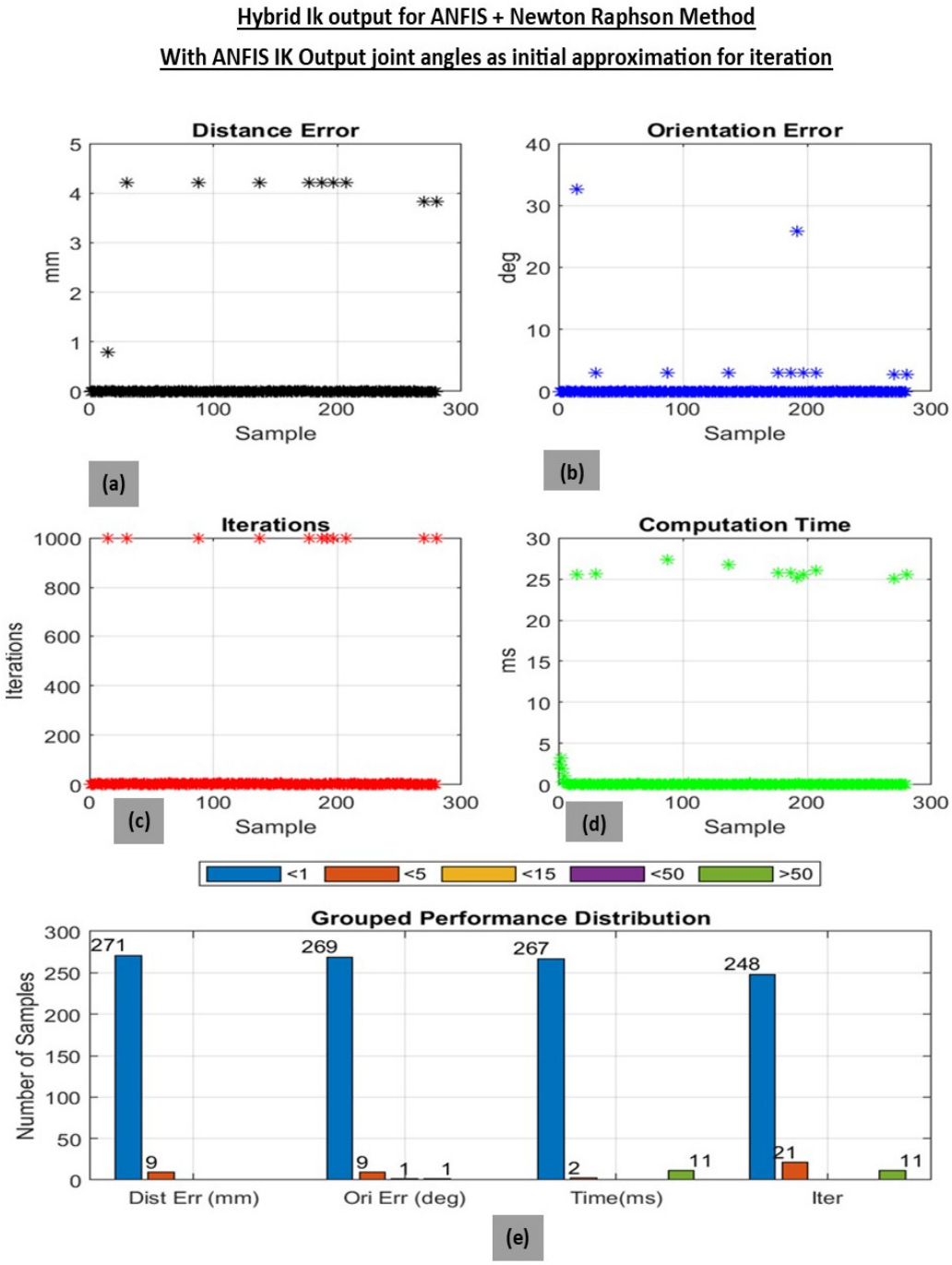


Figure 13. Hybrid IK (ANFIS + Newton–Raphson) results. (**a**) Distance Error, (**b**) Orientation Error, (**c**) No of iterations required, (**d**) Computation Time required, (**e**) Grouped performance distribution.

4.6. Damped Least Squares (DLS) Method

In Experiment 6, the Damped Least Squares (DLS) method was applied using the home configuration as the initial joint vector. The Damped Least Squares method is evaluated to analyze the behavior of a damped Jacobian-based solver within the proposed two-stage hybrid IK framework. Results obtained from Experiment 6 are shown in Figure 14. The DLS solver exhibits improved numerical stability compared to the pseudoinverse when initialized from the home configuration. However, Figure 14a shows that it still converges slowly and leaves large residual errors. Distance errors remain in the range of 300–600 mm, orientation errors remain high, and iteration counts approach 500, indicating inefficient convergence from a distant initial state.

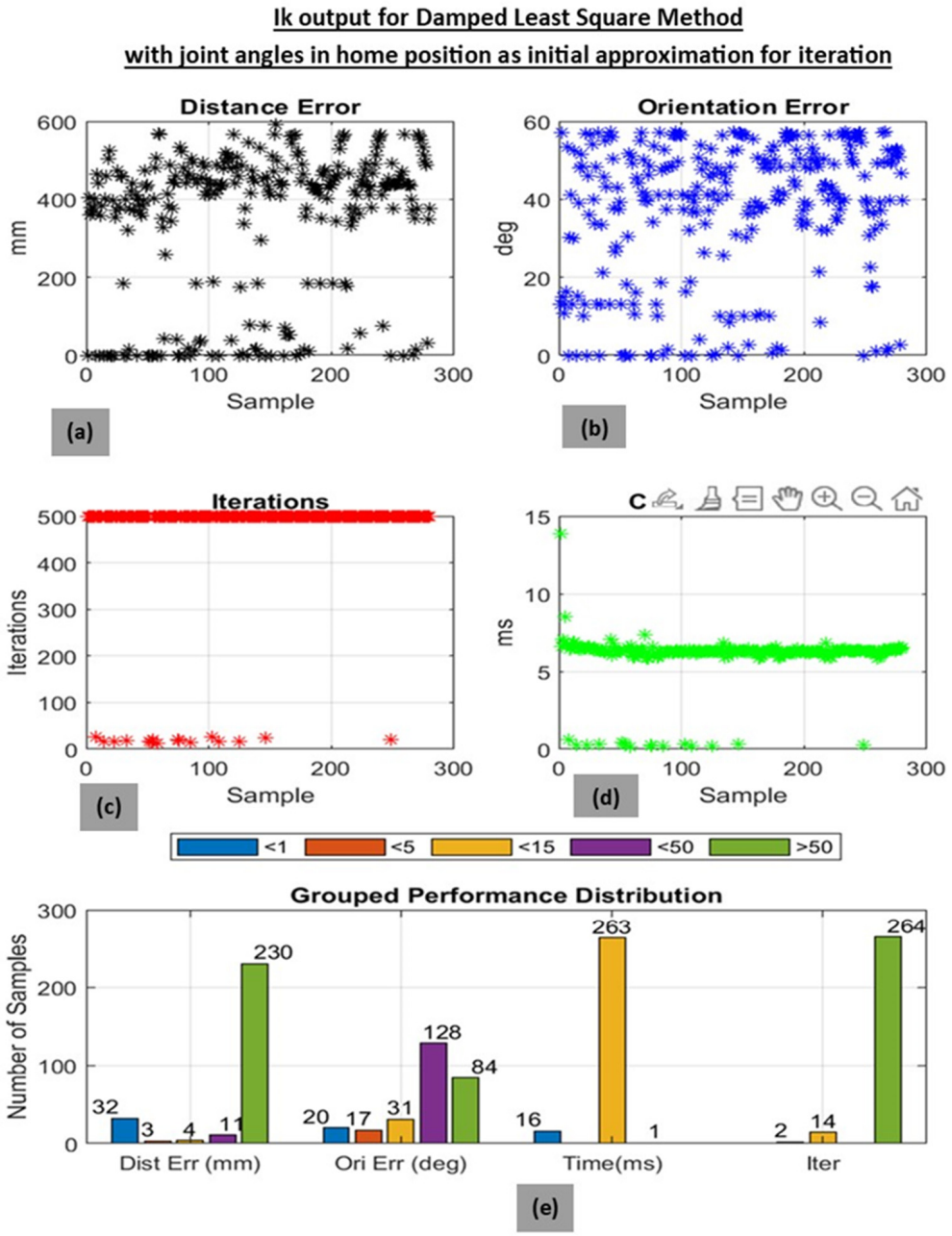


Figure 14. IK results obtained for Damped Least Squares method. (**a**) Distance Error, (**b**) Orientation Error, (**c**) No of iterations required, (**d**) Computation Time required, (**e**) Grouped performance distribution.

4.7. Hybrid Framework-2 (ANFIS + Damped Least Squares Method)

In Experiment 7, the DLS method was initialized using the ANFIS output from Experiment 2, forming the Hybrid ANFIS + DLS IK. Results obtained from Experiment 7 are shown in Figure 15. When initialized using ANFIS-predicted joint angles, the DLS solver achieves the best overall performance. As shown in Figure 15b, distance errors are nearly zero, orientation errors remain below 3°, iteration counts are typically below 10, and computation time remains consistently under 1 ms. The damping term ensures robustness near singular configurations, while ANFIS ensures that the solver begins in close proximity to the true solution.

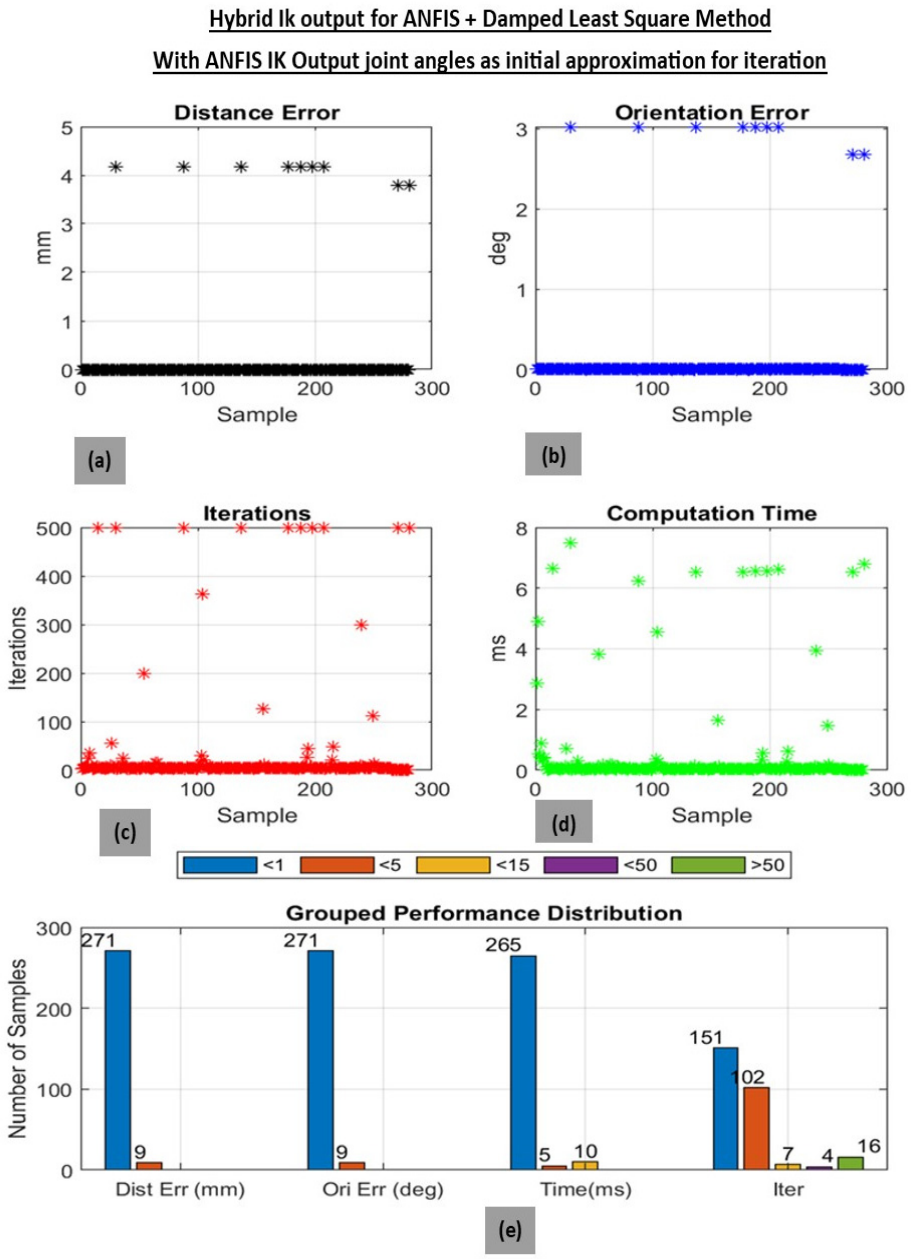


Figure 15. Hybrid IK (ANFIS + Damped Least Squares) results. (**a**) Distance Error, (**b**) Orientation Error, (**c**) No of iterations required, (**d**) Computation Time required, (**e**) Grouped performance distribution.

4.8. Comparative Interpretation

A consistent trend is observed across all three iterative solvers: the initialization strategy strongly governs convergence behavior. When initialized from the home position, the Gradient method stagnates, the Pseudoinverse method diverges, and the DLS method converges slowly with large residual errors. In contrast, when ANFIS provides the initial joint configuration, all three solvers exhibit rapid convergence, low position and orientation errors, and stable computational performance.

Among the three hybrid combinations, ANFIS + DLS delivers the most reliable and robust performance, providing high accuracy with low iteration counts and strong numerical stability. ANFIS + Pseudoinverse achieves the fastest convergence when well-conditioned, while ANFIS + Gradient offers a computationally lightweight alternative with moderate accuracy.

A significant advantage of the Damped Least Squares (DLS) hybrid framework (Framework-3) is its behavior near singularities. In standard IK, the robots either freeze or jerk near a singularity point; however, the DLS method introduces a damping factor that maintains numerical stability, allowing for smooth, continuous motion even when the robot is near its physical joint limits or singular configurations.

5. Conclusions and Future Works

This paper presented a hybrid inverse kinematics (IK) framework that integrates a data-driven Adaptive Neuro-Fuzzy Inference System (ANFIS) with physics-based Jacobian iterative solvers to achieve fast, accurate, and robust IK solutions for industrial robotic manipulators. Unlike conventional approaches that rely solely on either analytical or numerical techniques, the proposed method exploits the complementary strengths of learning-based global approximation and model-based local refinement.

Experimental results demonstrated that ANFIS alone provides only a coarse IK solution, with residual position and orientation errors that are unsuitable for precision robotic manipulation. However, ANFIS reliably places the joint configuration within a narrow neighborhood of the true solution. When this estimate is used to initialize Jacobian-based solvers, convergence behavior improves dramatically across all tested methods.

The comparative evaluation of Jacobian Gradient, Jacobian Pseudoinverse, and Damped Least Squares (DLS) under both home-initialized and ANFIS-initialized conditions clearly showed that initialization dominates solver performance. Home-initialized solvers either stagnated, diverged, or converged slowly with large residual errors, whereas ANFIS-initialized solvers consistently achieved rapid convergence, low position and orientation errors, and stable computational cost.

Among the tested hybrid strategies, ANFIS + DLS emerged as the most reliable and robust solver, combining fast convergence with numerical stability near singularities. ANFIS + Pseudoinverse provided the fastest convergence in well-conditioned regions, while ANFIS + Gradient offered a computationally lightweight alternative for moderately constrained tasks. These results validate the effectiveness of the proposed hybrid IK paradigm for high-precision industrial robot control.

The evaluation of the proposed algorithm on the 6-DoF ABB IRB 1200 serves as a proof of concept that can be scaled to accommodate any N-DoF serial revolute configuration. In future work, we intend to apply this to various robotic arms and discuss the results.

This preliminary research opens a new path for using structured datasets in robotics. In future studies, we intend to develop a Digital Twin to simulate the robot using real-time data, ensuring perfect synchronization between the virtual and physical systems.

In future work, we intend to investigate quaternion-based orientation representation to study the stability and smoothness of learning and numerical refinement beyond the current Z–Y–X Euler-angle implementation.

Author Contributions: Conceptualization, M.J.; Supervision, K.L. and P.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

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

Appendix A. Forward Kinematics of the Robotic Arm

The forward kinematic model establishes the relationship between the joint variables and the resulting end-effector pose. Once the Denavit–Hartenberg (DH) parameters are defined, the homogeneous transformation matrix of each link is constructed by sequentially transforming the coordinate frame from joint i to joint i + 1. Using the standard DH formulation, the transformation matrix is expressed as

$$ {}^{i-1}T_i = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where $c\theta_i$ is cos θ_i, $s\theta_i$ is sin θ_i, $c\alpha_i$ is cos α_i, $s\alpha_i$ is sin α_i and i = 1 to 6;

$$ {}^{0}\mathrm{T}_6 = {}^{0}\mathrm{T}_1 * {}^{1}\mathrm{T}_2 * {}^{2}\mathrm{T}_3 * {}^{3}\mathrm{T}_4 * {}^{4}\mathrm{T}_5 * {}^{5}\mathrm{T}_6 $$

$$ {}^{0}\mathrm{T}_6 = \begin{bmatrix} n_x & s_x & a_x & d_x \\ n_y & s_y & a_y & d_y \\ n_z & s_z & a_z & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Let $\cos(\theta_i) = c_i$, $\sin(\theta_i) = s_i$,

$$ {}^{0}\mathrm{T}_1 = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ s_1 & 0 & -c_1 & 0 \\ 0 & 1 & 0 & \mathrm{L}_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

${}^{1}\mathrm{T}_2$ is the transformation of frame-2 to frame-1,

$$ {}^{1}\mathrm{T}_2 = \begin{bmatrix} c_2 & 0 & s_2 & 0 \\ s_2 & 0 & -c_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

${}^{2}\mathrm{T}_3$ is the transformation of frame-3 to frame-2,

$$ {}^{2}\mathrm{T}_3 = \begin{bmatrix} c_3 & 0 & s_3 & 0 \\ s_3 & 0 & -c_3 & 0 \\ 0 & 1 & 0 & -\mathrm{L}_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

${}^{3}\mathrm{T}_4$ is the transformation of frame-4 to frame-3,

$$ {}^{3}\mathrm{T}_4 = \begin{bmatrix} c_4 & 0 & s_4 & 0 \\ s_4 & 0 & -c_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$^{4}T_{5}$ is the transformation of frame-5 to frame-4,

$$^{4}T_{5} = \begin{bmatrix} c_5 & 0 & s_5 & 0 \\ s_5 & 0 & -c_5 & 0 \\ 0 & 1 & 0 & L_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$^{5}T_{6}$ is the transformation of frame-6 to frame-5,

$$^{5}T_{6} = \begin{bmatrix} -s_6 & 0 & c_6 & 0 \\ c_6 & 0 & s_6 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$n_x = -\sin(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) - \cos(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)))$$

$$s_x = \sin(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3))) - \cos(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2)))$$

$$a_x = \cos(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)) - \sin(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2)))$$

$$d_x = 448*\cos(q1)*\sin(q2) - 82*\sin(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + 82*\cos(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)) + 451*\cos(q1)*\cos(q2)*\cos(q3) + 42*\cos(q1)*\cos(q2)*\sin(q3) + 42*\cos(q1)*\cos(q3)*\sin(q2) - 451*\cos(q1)*\sin(q2)*\sin(q3)$$

$$n_y = -\sin(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) - \cos(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)))$$

$$s_y = -\sin(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) - \cos(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)))$$

$$a_y = -\sin(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) - \cos(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)))$$

$$d_y = 448*\sin(q1)*\sin(q2) + 82*\sin(q5)*(\cos(q1)*\sin(q4) - \cos(q4)*(\cos(q2)*\sin(q1)*\sin(q3) + \cos(q3)*\sin(q1)*\sin(q2))) - 82*\cos(q5)*(\sin(q1)*\sin(q2)*\sin(q3) - \cos(q2)*\cos(q3)*\sin(q1)) - 451*\sin(q1)*\sin(q2)*\sin(q3) + 451*\cos(q2)*\cos(q3)*\sin(q1) + 42*\cos(q2)*\sin(q1)*\sin(q3) + 42*\cos(q3)*\sin(q1)*\sin(q2)$$

$$n_z = -\sin(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) - \cos(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)))$$

$$s_z = -\sin(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) - \cos(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)))$$

$$a_z = -\sin(q6)*(\cos(q4)*\sin(q1) - \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) - \cos(q6)*(\cos(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2))) + \sin(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)))$$

$$d_z = 448*\cos(q2) + 42*\cos(q2)*\cos(q3) - 451*\cos(q2)*\sin(q3) - 451*\cos(q3)*\sin(q2) - 42*\sin(q2)*\sin(q3) - 82*\cos(q5)*(\cos(q2)*\sin(q3) + \cos(q3)*\sin(q2)) - 82*\cos(q4)*\sin(q5)*(\cos(q2)*\cos(q3) - \sin(q2)*\sin(q3)) + 3991/10$$

Appendix B. Jacobian Matrix of the Robotic Arm

$$J_{11} = 82*\sin(q5)*(\cos(q2)*\cos(q4)*\sin(q1)*\sin(q3) - \cos(q1)*\sin(q4) + \cos(q3)*\cos(q4)*\sin(q1)*\sin(q2)) - 448*\sin(q1)*\sin(q2) + 451*\sin(q1)*\sin(q2)*\sin(q3) - 82*\cos(q2 + q3)*\cos(q5)*\sin(q1) - 451*\cos(q2)*\cos(q3)*\sin(q1) - 42*\cos(q2)*\sin(q1)*\sin(q3) - 42*\cos(q3)*\sin(q1)*\sin(q2)$$

$$J_{12} = \cos(q1)*(42*\cos(q2 + q3) - 451*\sin(q2 + q3) + 448*\cos(q2) + 41*\sin(q4 - q5)*\cos(q2 + q3) - 41*\cos(q2 + q3)*\sin(q4 + q5) - 82*\sin(q2 + q3)*\cos(q5))$$

$$J_{13} = -\cos(q1)*(451*\sin(q2 + q3) - 42*\cos(q2 + q3) + 82*\sin(q2 + q3)*\cos(q5) + 82*\cos(q2 + q3)*\cos(q4)*\sin(q5))$$

$$J_{14} = 82*\sin(q5)*(\cos(q1)*\cos(q2)*\sin(q3)*\sin(q4) - \cos(q4)*\sin(q1) + \cos(q1)*\cos(q3)*\sin(q2)*\sin(q4))$$

$$J_{15} = 82*\cos(q1)*\sin(q2)*\sin(q3)*\sin(q5) - 82*\cos(q1)*\cos(q2)*\cos(q3)*\sin(q5) - 82*\cos(q5)*\sin(q1)*\sin(q4) - 82*\cos(q1)*\cos(q2)*\cos(q4)*\cos(q5)*\sin(q3) - 82*\cos(q1)*\cos(q3)*\cos(q4)*\cos(q5)*\sin(q2)$$

$$J_{16} = 0$$

$$J_{21} = 448*\cos(q1)*\sin(q2) - 82*\sin(q5)*(\sin(q1)*\sin(q4) + \cos(q1)*\cos(q2)*\cos(q4)*\sin(q3) + \cos(q1)*\cos(q3)*\cos(q4)*\sin(q2)) + 82*\cos(q2 + q3)*\cos(q1)*\cos(q5) + 451*\cos(q1)*\cos(q2)*\cos(q3) + 42*\cos(q1)*\cos(q2)*\sin(q3) + 42*\cos(q1)*\cos(q3)*\sin(q2) - 451*\cos(q1)*\sin(q2)*\sin(q3)$$

$$J_{22} = \sin(q1)*(42*\cos(q2 + q3) - 451*\sin(q2 + q3) + 448*\cos(q2) + 41*\sin(q4 - q5)*\cos(q2 + q3) - 41*\cos(q2 + q3)*\sin(q4 + q5) - 82*\sin(q2 + q3)*\cos(q5))$$

$$J_{23} = -\sin(q1)*(451*\sin(q2 + q3) - 42*\cos(q2 + q3) + 82*\sin(q2 + q3)*\cos(q5) + 82*\cos(q2 + q3)*\cos(q4)*\sin(q5))$$

$$J_{24} = 82*\sin(q5)*(\cos(q1)*\cos(q4) + \cos(q2)*\sin(q1)*\sin(q3)*\sin(q4) + \cos(q3)*\sin(q1)*\sin(q2)*\sin(q4))$$

$$J_{25} = 82*\cos(q1)*\cos(q5)*\sin(q4) - 82*\cos(q2)*\cos(q3)*\sin(q1)*\sin(q5) + 82*\sin(q1)*\sin(q2)*\sin(q3)*\sin(q5) - 82*\cos(q2)*\cos(q4)*\cos(q5)*\sin(q1)*\sin(q3) - 82*\cos(q3)*\cos(q4)*\cos(q5)*\sin(q1)*\sin(q2)$$

$$J_{26} = 0$$

$$J_{31} = 0$$

$$J_{32} = 451*\sin(q2)*\sin(q3) - 451*\cos(q2)*\cos(q3) - 42*\cos(q2)*\sin(q3) - 42*\cos(q3)*\sin(q2) - 448*\sin(q2) - 82*\cos(q2)*\cos(q3)*\cos(q5) + 82*\cos(q5)*\sin(q2)*\sin(q3) + 82*\cos(q2)*\cos(q4)*\sin(q3)*\sin(q5) + 82*\cos(q3)*\cos(q4)*\sin(q2)*\sin(q5)$$

$$J_{33} = 451*\sin(q2)*\sin(q3) - 42*\cos(q2)*\sin(q3) - 42*\cos(q3)*\sin(q2) - 451*\cos(q2)*\cos(q3) - 82*\cos(q2)*\cos(q3)*\cos(q5) + 82*\cos(q5)*\sin(q2)*\sin(q3) + 82*\cos(q2)*\cos(q4)*\sin(q3)*\sin(q5) + 82*\cos(q3)*\cos(q4)*\sin(q2)*\sin(q5)$$

$$J_{34} = 82*\cos(q2 + q3)*\sin(q4)*\sin(q5)$$

$$J_{35} = 82*\cos(q2)*\sin(q3)*\sin(q5) + 82*\cos(q3)*\sin(q2)*\sin(q5) - 82*\cos(q2)*\cos(q3)*\cos(q4)*\cos(q5) + 82*\cos(q4)*\cos(q5)*\sin(q2)*\sin(q3)$$

$$J_{36} = 0$$

$$J_{41} = 0$$

$$J_{42} = -\sin(q1)$$

$$J_{43} = -\sin(q1)$$

$$J_{44} = \cos(q2 + q3)*\cos(q1)$$

$$J_{45} = \sin(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2)) - \cos(q4)*\sin(q1)$$

$$J_{46} = \cos(q5)*(\cos(q1)*\cos(q2)*\cos(q3) - \cos(q1)*\sin(q2)*\sin(q3)) - \sin(q5)*(\sin(q1)*\sin(q4) + \cos(q4)*(\cos(q1)*\cos(q2)*\sin(q3) + \cos(q1)*\cos(q3)*\sin(q2)))$$

$$J_{51} = 0$$

$$J_{52} = \cos(q1)$$

$$J_{53} = \cos(q1)$$

$$J_{54} = \cos(q2 + q3)*\sin(q1)$$

$$J_{55} = \cos(q1)*\cos(q4) + \sin(q4)*(\cos(q2)*\sin(q1)*\sin(q3) + \cos(q3)*\sin(q1)*\sin(q2))$$

$$J_{56} = \sin(q5)*(\cos(q1)*\sin(q4) - \cos(q4)*(\cos(q2)*\sin(q1)*\sin(q3) + \cos(q3)*\sin(q1)*\sin(q2))) - \cos(q5)*(\sin(q1)*\sin(q2)*\sin(q3) - \cos(q2)*\cos(q3)*\sin(q1))$$

$$J_{61} = 1$$

$$J_{62} = 0$$

$$J_{63} = 0$$

$$J_{64} = -\sin(q2 + q3)$$

$$J_{65} = \cos(q2 + q3)*\sin(q4)$$

$$J_{66} = -\sin(q2 + q3)*\cos(q5) - \cos(q2 + q3)*\cos(q4)*\sin(q5)$$

References

1. Kircanski, M.; Tatjana, T.M. Analytical solutions for non-redundant manipulators. *IEEE Trans. Syst. Man Cybern.* **1991**, *21*, 327–335. [CrossRef]
2. Song, D.H.; Jung, S. Geometrical Analysis of Inverse Kinematics Solutions and Fuzzy Control of Humanoid Robot Arm Under Kinematics Constraints. In Proceedings of the 2007 International Conference on Mechatronics and Automation, Harbin, China, 5–8 August 2007; pp. 1178–1183. [CrossRef]
3. Carbajal-Espinosa, O.; Campos-Macías, L.; Díaz-Rodriguez, M. FIKA: A Conformal Geometric Algebra Approach to a Fast Inverse Kinematics Algorithm for an Anthropomorphic Robotic Arm. *Machines* **2024**, *12*, 78. [CrossRef]
4. Carbonari, L.; Palpacelli, M.-C.; Callegari, M. Inverse Kinematics of a Class of 6R Collaborative Robots with Non-Spherical Wrist. *Robotics* **2023**, *12*, 36. [CrossRef]
5. Ahmed, A.; Omar, M.; Ali, F. Unified analytical–numerical IK formulation using quaternion mathematics. *Mathematics* **2025**, *13*, 25. [CrossRef]
6. Whitney, D.E. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man Mach. Syst.* **1969**, *10*, 47–53. [CrossRef]
7. Klein, C.A.; Huang, C.H. Review of pseudoinverse control for use with kinematically redundant manipulators. *IEEE Trans. Syst. Man Cybern.* **2012**, *2*, 245–250. [CrossRef]
8. Wampler, C.W. Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods. *IEEE Trans. Syst. Man Cybern.* **1986**, *16*, 93–101. [CrossRef]
9. Anschober, J.; Schmied, T.; Zangl, H. Efficient embedded-friendly inverse kinematics for 6R robot arms. *Sensors* **2023**, *23*, 2389. [CrossRef]
10. Shi, X.; Guo, Y.; Chen, X.; Chen, Z.; Yang, Z. Kinematics and Singularity Analysis of a 7-DOF Redundant Manipulator. *Sensors* **2021**, *21*, 7257. [CrossRef]
11. Aristidou, A.; Lasenby, J. FABRIK: A fast, iterative solver for the inverse kinematics problem. *Graph. Model.* **2011**, *73*, 243–260. [CrossRef]
12. Kenwright, B. Real-time inverse kinematics using the Gauss–Seidel iterative approximation. *Int. J. Comput. Graph. Anim.* **1986**, *2*, 32–39.
13. Aristidou, A.; Lasenby, J. *Iterative Inverse Kinematics: A Review and Fast Solver*; CUED/F-INFENG/TR-632; University of Cambridge: Cambridge, UK, 2009.
14. Poddighe, R. Comparing FABRIK and neural networks for inverse kinematics. *Comput. Graph. Forum* **2013**, *32*, 174–184. [CrossRef]
15. Li, M.; Luo, X.; Qiao, L. Inverse Kinematics of Robot Manipulator Based on BODE-CS Algorithm. *Machines* **2023**, *11*, 648. [CrossRef]
16. Zhao, G.; Tao, B.; Jiang, D.; Yun, J.; Fan, H. Improved Bald Eagle Search Optimization Algorithm for the Inverse Kinematics of Robotic Manipulators. *Biomimetics* **2024**, *9*, 627. [CrossRef]
17. Huang, Y.; Liu, J.; Zhang, X.; Wang, J.; Li, X.; Tu, X.; Chen, S.; Wang, C.; Huang, Q. An Efficient Computational Approach for Inverse Kinematics Analysis of the UR10 Robot with SQP and BP-SQP Algorithms. *Appl. Sci.* **2023**, *13*, 3009. [CrossRef]
18. Hernandez-Barragan, J.; Martinez-Soltero, G.; Rios, J.D.; Lopez-Franco, C.; Alanis, A.Y. A Metaheuristic Optimization Approach to Solve Inverse Kinematics of Mobile Dual-Arm Robots. *Mathematics* **2022**, *10*, 4135. [CrossRef]
19. Monfared, P.; Moosavian, A.; Vafadar, F. PSO-ANN-based hybrid inverse kinematics for redundant manipulators. *Eng. Proc.* **2024**, *27*, 72.
20. Lan, L.; Xu, Z.; Yu, D. Manipulator kinematics learning using vector quantized topological associative memory (VQTAM). *Symmetry* **2020**, *12*, 1110. [CrossRef]
21. Kramar, V.; Kramar, O.; Kabanov, A. An Artificial Neural Network Approach for Solving Inverse Kinematics Problem for an Anthropomorphic Manipulator of Robot SAR-401. *Machines* **2022**, *10*, 241. [CrossRef]
22. Iqdymat, A.; Stamatescu, G. Reinforcement learning for pick-and-place inverse kinematics of industrial manipulators. *Sustainability* **2025**, *17*, 622. [CrossRef]
23. Ibarra-Pérez, T.; Ortiz-Rodríguez, J.M.; Olivera-Domingo, F.; Guerrero-Osuna, H.A.; Gamboa-Rosales, H.; Martínez-Blanco, M.d.R. A Novel Inverse Kinematic Solution of a Six-DOF Robot Using Neural Networks Based on the Taguchi Optimization Technique. *Appl. Sci.* **2022**, *12*, 9512. [CrossRef]

24. Tagliani, F.L.; Pellegrini, N.; Aggogeri, F. Machine Learning Sequential Methodology for Robot Inverse Kinematic Modelling. *Appl. Sci.* **2022**, *12*, 9417. [CrossRef]
25. Bai, Y.; Luo, M.; Pang, F. An Algorithm for Solving Robot Inverse Kinematics Based on FOA Optimized BP Neural Network. *Appl. Sci.* **2021**, *11*, 7129. [CrossRef]
26. Lee, C.; An, D. AI-Based Posture Control Algorithm for a 7-DOF Robot Manipulator. *Machines* **2022**, *10*, 651. [CrossRef]
27. Dharmana, M.; Kumar, A.; Suresh, K. Embedded ANFIS controller for 6-DOF robotic manipulators. *Procedia Comput. Sci.* **2014**, *47*, 168–175. [CrossRef]
28. Singh, E.; Verma, P.; Yadav, R. ANFIS-based forward and inverse kinematics for 6-DOF robot arms. *Int. J. Mech. Eng. Robot. Res.* **2015**, *4*, 309–316.
29. Hendarto, H. Adaptive neuro-fuzzy inverse kinematics of PUMA 560 robot. *Procedia Technol.* **2014**, *11*, 963–970. [CrossRef]
30. Jasim, W.M. Inverse kinematics for SCARA robots using ANFIS. *Int. J. Soft Comput.* **2011**, *6*, 63–69.
31. Hernandez-Barragan, J.; Plascencia-Lopez, J.; Lopez-Franco, M.; Arana-Daniel, N.; Lopez-Franco, C. Inverse Kinematics of Robotic Manipulators Based on Hybrid Differential Evolution and Jacobian Pseudoinverse Approach. *Algorithms* **2024**, *17*, 454. [CrossRef]
32. ABB ABB Group. IRB 1200 Next Generation. Available online: https://www.abb.com/global/en/areas/robotics/products/robots/articulated-robots/small-robots/irb-1200-next-generation (accessed on 7 December 2025).
33. ABB. Product Specification—IRB 1200. 2025. Available online: https://library.e.abb.com/public/a99430c5e3204420b733d9bda60b318b/3HAC081417%20PS%20IRB%201200%20on%20OmniCore-en.pdf?x-sign=BJwvVT5sukGOCCu1FnuahuKg3hvOVkLSiZYPfGHELgfZ26h9iTmvb29sWgQrESZr (accessed on 7 December 2025).
34. Krishnan, M.G.; Ashok, S. Kinematic Analysis and Validation of an Industrial Robot Manipulator. In Proceedings of the ENCON 2019-2019 IEEE Region 10 Conference (TENCON), Kochi, India, 17–20 October 2019; Volume 19, pp. 1393–1399. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.