FORWARD AND INVERSE KINEMATICS OF THE MOTOMAN HP3 MANIPULATOR

Osama A. Abolaeha[1,2], Ahmed J. Abougarair[1], and Mohamed K. Aburakhis[1,3]

[1]Electrical & Electronics Engineering, University of Tripoli, Libya
[2]Embedded Software Developer Stulz Digital Solutions, Germany
[3]Valparaiso University, Indiana, Valparaiso, USA
Email: a.abougarair@uot.edu.ly
Received 28 August 2025; Revised 13 October 2025; Accepted 15 October 2025; Published 10 November 2025

الملخص

تقدم هذه الورقة تحليلاً شاملاً للحركيات الأمامية والعكسية لـ MOTOMAN HP3 ، وهو روبوت مُعالج بست درجات حرية. تُشتقّ الحركات الأمامية باستخدام خوارزمية Denavit-Hartenberg (DH) لحساب موضع واتجاه المُستقبِل النهائي من زوايا المفصل. طُوِّرت واجهة مستخدم رسومية MATLAB لتنفيذ نموذج الحركيات الأمامية والتحقق من صحته. بالنسبة للحركات العكسية، يُقترح نهج هجين يجمع بين الأساليب التحليلية والشبكات العصبية الاصطناعية ذات الدالة الشعاعية الأساسية (RBF-ANN) لمعالجة التعقيد الحسابي والتفردات. تُحل زاوية المفصل الأولى هندسيًا، بينما تُقَرَّب الزاويتان الثانية والثالثة باستخدامRBF-ANN ، وتُحسب الزوايا المتبقية تحليليًا. تُثبت النتائج التجريبية دقة كلا النموذجين مقارنةً ببرنامج محاكاة MOTOMAN، مما يُظهر فعاليتهما في التطبيقات الروبوتية.

ABSTRACT

This paper provides a detailed study of the forward and inverse kinematics of the MOTOMAN HP3, a 6-degree-of-freedom robotic manipulator. The forward kinematics are derived using the Denavit–Hartenberg (DH) method to calculate the end-effector's position and orientation from the joint angles, and a MATLAB GUI is developed to implement and validate the model. For inverse kinematics, a hybrid approach is proposed that combines analytical methods with Radial Basis Function Artificial Neural Networks (RBF-ANN) to overcome computational complexity and handle singularities. The first joint angle is determined geometrically, the second and third angles are approximated via RBF-ANN, and the remaining angles are computed analytically. Experimental results confirm the accuracy of both models when compared with the MOTOMAN simulation software, demonstrating their effectiveness for robotic applications.

KEYWORDS: Forward Kinematics, Inverse Kinematics, HP3 Manipulator, Denavit-Hartenberg, MATLAB GUI, RBF-ANN.

INTRODUCTION

Kinematics examines the movement of objects without accounting for the forces or torques responsible for that motion. In robotics, this involves the mathematical examination of a manipulator's movement. Developing appropriate kinematic models is essential for evaluating the performance of industrial robotic arms. Primarily includes forward kinematics (FK) and inverse kinematics (IK). This end-effector's location and pose based on the inverse kinematics (IK). FK influences the end-effector's location and pose based on the manipulator's joint variables. This technique finds applications in robotics, gaming, and computer animation. In contrast, IK presents greater challenges, as

it requires determining joint configurations given the end-effector's desired position within the workspace. The literature highlights significant progress in IK techniques, emphasizing their precision, speed, and suitability for arms like the MOTOMAN HP3. Contemporary studies have investigated combined strategies to optimize accuracy alongside real-time execution. Table (1) offers a critical summary of these methods, identifying limitations in managing multi-degree-of-freedom systems and variable environments. Positioning our integrated (analytical + RBF-ANN) technique amid these developments highlights its innovation in tackling the HP3's unique issues, including interdependent axes and motion constraints.

Table 1: Summarized abstract of literature review

Ref.	Summarized Abstract
[1]	Exploration of IK challenges for robotic arms.
[2]	Industrial robots execute operations in structured settings.
[3]	Utilization of MLFFNN for FK and IK in a 3-DOF arm.
[4]	MLFFNN applied to manipulator FK and IK. Dual scenarios for FK: focusing on coordinates or full pose.
[5]	Emphasis on both FK and IK formulations.
[6]	Integration of analytical and ANFIS approaches for motion evaluation.
[7]	Offline-trained adaptive NN using LQOSEIC for initial weights, then online adaptation via error feedback.
[8]	Emphasis on open-chain kinematic evaluation. T-Matrix technique for tracking terminal link displacement.
[9]	VAE framework covering FK, IK, and redundancy resolution.
[10]	Design of a 5-DOF wheeled robot enhancing operational range.
[11]	Provision of FK and IK algorithms and - Evaluation of arm's operational volume.
[12]	Handling of FK and IK for 3-PSP structures.
[13]	Closed-form FK for 3-RPR configurations.
[14]	3RRR parallel arm modeling via GA and NN.
[15]	PID-LQR hybrid controller for TWRM, enhancing response and steadiness.
[16]	This study presents a robotic arm control system based on surface electromyography (EMG) signals from forearm muscles.
[17]	Generalized approach for 3RPS and 3RPS-R manipulators' kinematics.
[18]	AMPC system dynamically adjusts control parameters using sensor fusion data in a 3-DOF bicycle model.
[19]	Examination of arm and structural kinematics and - Analytical closures for diverse layouts.
[20]	Vision-based robotic arm control using PD-PIJ kinematics.
[21]	Kinematic analysis and simulation using DH parameters.
[22]	Closed-form inverse kinematics for 5-DOF hybrid manipulator without geometry assumptions.
[23]	Surrogate model using VQTAM + K-means for IK. Combines LLR, LWR, and LLE for improved prediction.
[24]	Forward/inverse kinematics analyzed for 5-DOF manipulator using MATLAB.
[25]	Review comparing different kinematic modeling techniques.
[26]	Hybrid intelligent control using LQRWFPI and supervised neural networks for nonlinear systems.
[27]	Robust robot control via SMC and Parallel PID-LQR strategies.

This study details the FK and IK derivations for the 6-DOF MOTOMAN HP3 arm. The structure proceeds as follows: An initial overview of the HP3 arm, followed by FK and IK sections. The FK part begins with homogeneous transformations, then covers the DH method, derives HP3 matrices, and includes angle-to-pulse conversions. A MATLAB GUI supports this. Validation occurs via experiments. The IK section uses a blended analytical-numerical strategy: geometric solution for θ1, RBF-ANN for θ2/θ3, and analytical for θ4-θ6. Results confirm efficacy. The conclusion summarizes and suggests extensions [4].

THE HP3 MANIPULATOR

The MOTOMAN HP3 represents a compact, rapid-response robotic arm ideal for space-constrained setups. As illustrated in Figure (1), it achieves a 701 mm arm span, maximizing its operational area relative to peers. Its versatile mounting options—floor, wall, or overhead—enhance adaptability. Optimized for precision tasks like component assembly, fluid application, packing, transport, and equipment oversight, the HP3 delivers high throughput with low setup costs. Notably, Figure (1) reveals its six rotational joints [28].

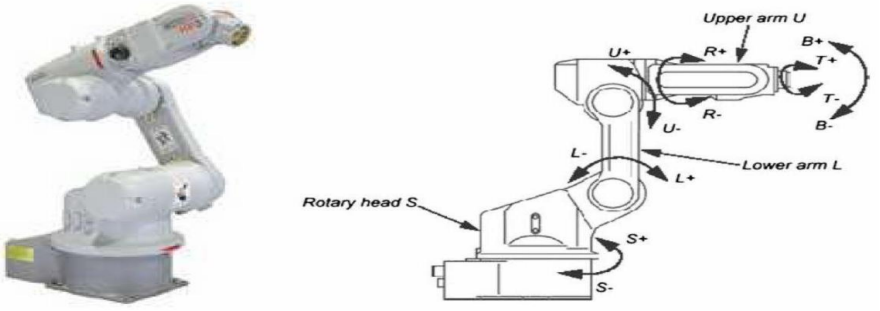

Figure 1: The MOTOMAN HP3 Manipulator.

Advantages Over Competing Robots

Best reach-to-size ratio in its class (701 mm in a compact form). Faster cycle times than comparable 3 kg payload robots. Multi-mounting flexibility (floor, wall, ceiling). Seamless integration with vision systems, force sensors, and IoT (Industry 4.0). Table (2) presents the key attributes of the MOTOMAN HP3 arm.

Table 2: Key attributes of the MOTOMAN HP3 arm.

Parameter	Specification
Robot Type	6-axis articulated (J1-J6, all revolute)
Payload Capacity	3 kg (maximum at full extension)
Repeatability	±0.02 mm (extremely precise)
Weight	~ 27 kg (lightweight for easy integration)
IP Rating	Standard: IP30 (optional IP67 for harsh environments)
Controller	Compatible with Yaskawa DX100 or YRC1000

FORWARD/DIRECT AND INVERSE KINEMATICS

As noted, manipulator kinematics splits into FK—straightforward with unique solutions—and IK, which is more demanding due to computational demands, singularities, and nonlinear effects. Complete closed-form IK exists only for select geometries. Figure (2) diagrams this interplay.

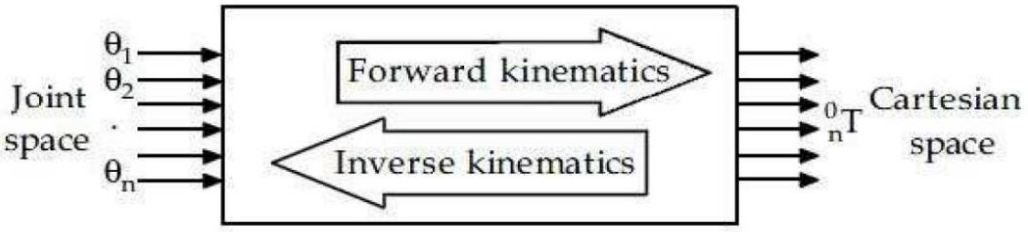

Figure 2: The diagram demonstration of forward and inverse kinematics.

Forward/Direct Kinematics

A robotic arm comprises interconnected segments joined by actuators. FK calculates the terminal link's pose from joint inputs (angles or pulses). This involves assigning frames to links and linking them via transformations.

Shifting between frames combines rotation and displacement. Common rotation forms include Euler angles, though homogeneous 4x4 matrices prevail in robotics for their compactness. Of these demonstrations, homogenous revolutions based on 4×4 actual matrices (orthonormal matrices) have been applied most frequently in robotics [28].

Let q be a point in $\mathbb{R}^3$, and let F is an orthonormal coordinate frame for $\mathbb{R}^3$. If σ is any nonzero scale factor, then the standardized coordinates of q with respect to F are denoted $[q]^F$ and defined as [29,30]:

$$[\boldsymbol{q}]^F = \sigma \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ 1 \end{bmatrix} \tag{1}$$

Note that the homogeneous coordinates of the point q are represented by a vector in fourdimensional space $\mathbb{R}^4$. Also, to note is that, in robotics, we usually manipulate a scale factor $\sigma = 1$ for convenience.

If a physical point in 3D space is stated in terms of its homogeneous coordinates and we want to change from one coordinate frame to another, we use a 4×4 homogeneous transformation matrix. In general, a homogeneous transformation matrix T can be partitioned into four separate sub-matrices as follows [30]:

$$T = \begin{bmatrix} R & P \\ \eta^T & \sigma \end{bmatrix} \tag{2}$$

Here, the value σ in he bottom-right position of the T matrix represents a non-zero $\sigma = 1$, and for the purpose of kinematics modeling the vector η will always be s always configured to the null vector [30].

The 3×3 sub-matrix R positioned in the top-left area of the matrix T functions as a rotation matrix [30]. R represents the orientation of the moveable coordinate frame with respect to the fixed reference frame [30]. For instance, a T corresponding to rotations about the x, y, or z axes by an angle θ are [31]:

$$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$Rot(y,\theta) = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$Rot(z,\theta) = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

The 3×1 column vector P located in the upper-right corner of matrix TTT, serves as the translation vector. It specifies the location of the moving coordinate frame's origin with respect to the fixed reference frame [30]. For instance, a transformation that translates by the vector $a\mathbf{i} + b\mathbf{j} + c\mathbf{k}$ can be expressed as [31]:

$$Tran(a,b,c) = \begin{bmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Denavit and Hartenberg Algorithm

FK relies on joint readings and arm geometry. The DH convention standardizes this with four parameters per link: The four parameters are:a_{i-1} (link length), α_{i-1} (link twist), d_i (link offset) and θ_i (joint angle) aligns with joint motion. As in Figure (3), ai along X_i, α_i around X_i, di along Z_{i-1}, θ_i around Z_{i-1} [32].

For each joint $i(i = 1,2, \dots, n)$. a coordinate frame is assigned to define the Denavit–Hartenberg (DH) parameters. The Z_i axis of each frame is oriented along the axis of rotation or translation of the corresponding joint. To clarify this concept, Figure (3) provides an example illustrating how coordinate frames are assigned for a general manipulator.

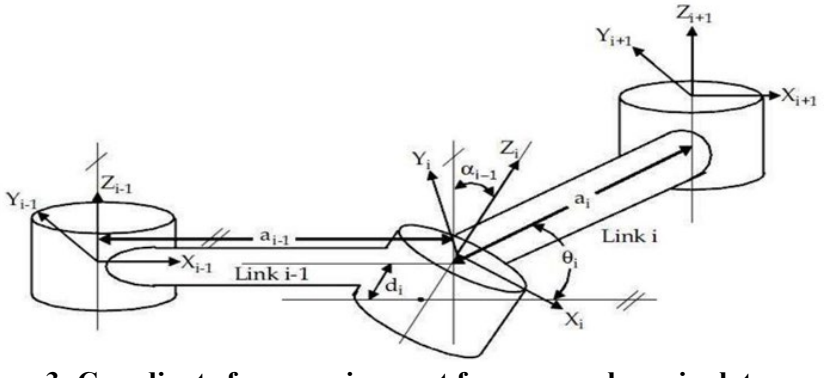

Figure 3: Coordinate frame assignment for a general manipulator.

As illustrated in Figure (3), the distance between Z_{i-1} and Z_i measured along the $X_{(i-1)}$ axis is denoted by a_{i-1}, the angle between Z_{i-1} and Z_i measured about the X_i axis s represented by α_{i-1}. Similarly, the distance from X_{i-1} to X_i measured along the Z_i axis is defined as d_i and the angle between X_{i-1} to X_i measured about Z_i is denoted by θ_i [32].

The general transformation matrix T_i^{i-1} for an individual link can then be expressed as follows:

$$T_i^{i-1} = Rot(x, \alpha_{i-1}) Tran(a_{i-1}, 0,0) Rot(z, \theta_i) Tran(0,0, d_i) \tag{7}$$

$$T_i^{i-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c(\alpha_{i-1}) & -s(\alpha_{i-1}) & 0 \\ 0 & s(\alpha_{i-1}) & c(\alpha_{i-1}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} c(\theta_i) & -s(\theta_i) & 0 & 0 \\ -s(\theta_i) & c(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_i^{i-1} = \begin{bmatrix} c(\theta_i) & -s(\theta_i) & 0 & a_{i-1} \\ s(\theta_i)c(\alpha_{i-1}) & c(\theta_i)c(\alpha_{i-1}) & -s(\alpha_{i-1}) & -s(\alpha_{i-1})d_i \\ s(\theta_i)s(\alpha_{i-1}) & c(\theta_i)s(\alpha_{i-1}) & c(\alpha_{i-1}) & c(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8}$$

Here $c(\theta_i)$ and $s(\theta_i)$ are abbreviations for $\cos(\theta_i)$ and $\sin(\theta_i)$, respectively. The forward kinematics of the end-effector relative to the base frame is obtained by taking the product of all individual transformation matrices T_i^{i-1} matrices [30-32]. In other words,

$$T_{base}^{end\text{-}effector} = T_1^0 T_2^1 \dots T_n^{n-1} \tag{9}$$

Forward Kinematics of the HP3 Manipulator Using the DH Algorithm

In this subsection, the required coordinate frames will be established, derive the DH parameters, and substitute the according values into the T matrices for the HP3 manipulator so as to obtain the general transformation matrix $T_{\text{base}}^{\text{end-effector}}$. To facilitate the calculation of the T matrices, we will form a table of joint and link parameters whereby the values representing each link and joint are determined from the schematic drawing of the robot, and are substituted in each T matrix. That being said, by simply inspecting the link frame assignment schematic drawing (shown in Figure (4)) and the various corresponding engineering dimensions of the HP3 manipulator illustrated in Figure (5) in millimeters, one can easily derive the four DH parameters/values for each link and joint using the DH method explained in next section [33].

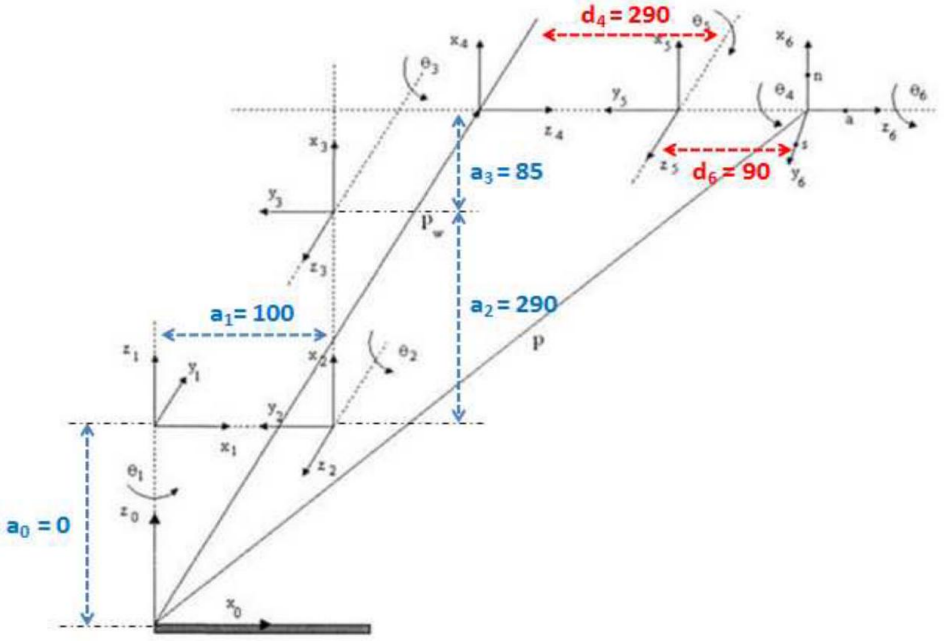

Figure 4: The HP3 Manipulator: link frame assignment schematic (for home position). Note that all dimensions are in millimeters.

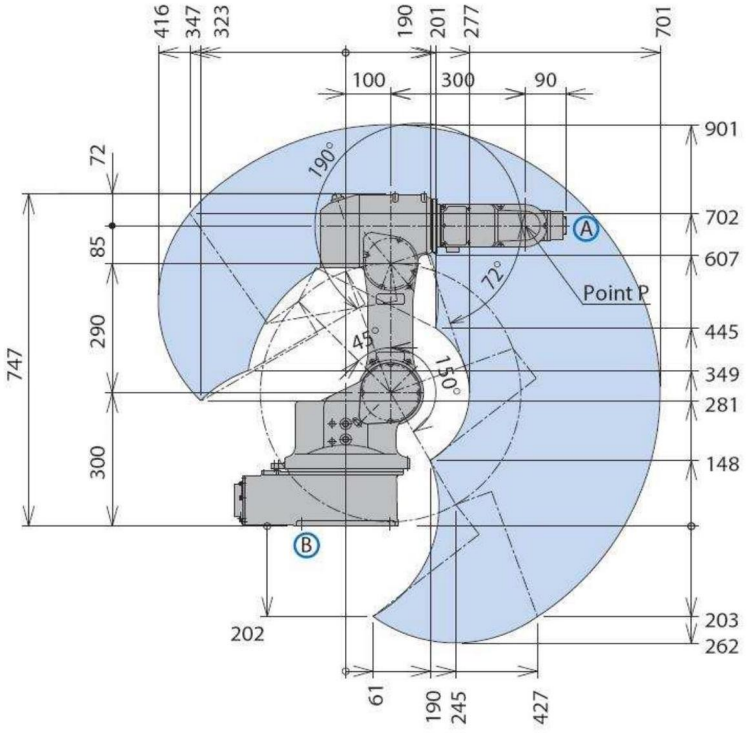

Figure 5: The HP3 Manipulator (home position): side-view with various engineering dimensions in millimeters.

The Denavit–Hartenberg (DH) parameters for each link of the HP3 manipulator are presented in Table (3), In this table θ_i is the joint angle, d_i denotes the joint offset, a_{i-1} s the link length, and α_{i-1} is the link twist. It is important to note that, for revolute joints (as in this case), $a_{i-1}, \alpha_{i-1}, d_i$ remain constant, while θ_i serves as the joint variable for

each joint $i(i = 1,2, \dots ,6)$. Based on the information depicted in Table (1) and using the general definition of the transformation matrix T_i^{i-1} has been derived, it becomes straightforward to calculate each link's transformation matrix as a function of its respective joint variable or angle (θ_i) as shown below:

$$T_1^0(\theta_1) = \begin{bmatrix} c(\theta_1) & -s(\theta_1) & 0 & 0 \\ s(\theta_1) & c(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

$$T_2^1(\theta_2) = \begin{bmatrix} c(\theta_2) & -s(\theta_2) & 0 & a_1 \\ s(\theta_2)c(\alpha_1) & c(\theta_2)c(\alpha_1) & -s(\alpha_1) & -s(\alpha_1)d_2 \\ s(\theta_2)s(\alpha_1) & c(\theta_2)s(\alpha_1) & c(\alpha_1) & c(\alpha_1)d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} c(\theta_2) & -s(\theta_2) & 0 & 100 \\ 0 & 0 & -1 & 0 \\ s(\theta_2) & c(\theta_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

Table 3: DH parameters for HP3 Manipulator (Note that all angles are in degrees and dimensions are in millimeters. Also, note that $\theta_2 = 90°$ portrays the home position of the manipulator).

Axis	Link(i)	θ_i	α_{i-1}	a_{i-1}	d_i	Joint Range	Max Speed
S	1	$\theta_1(0°)$	0°	0	0	170° to −170°	210°/s
L	2	$\theta_2(90°)$	90°	100	0	150° to −45°	180°/s
U	3	$\theta_3(0°)$	0°	290	0	210° to −142°	225°/s
R	4	$\theta_4(0°)$	90°	85	300	190° to −190°	375°/s
B	5	$\theta_5(0°)$	−90°	0	0	125° to −125°	375°/s
T	6	$\theta_6(0°)$	90°	0	90	360° to −360°	500°/s

$$T_3^2(\theta_3) = \begin{bmatrix} c(\theta_3) & -s(\theta_3) & 0 & a_2 \\ s(\theta_3)c(\alpha_2) & c(\theta_3)c(\alpha_2) & -s(\alpha_2) & -s(\alpha_2)d_3 \\ s(\theta_3)s(\alpha_2) & c(\theta_3)s(\alpha_2) & c(\alpha_2) & c(\alpha_2)d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c(\theta_3) & -s(\theta_3) & 0 & 290 \\ s(\theta_3) & c(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

$$T_4^3(\theta_4) = \begin{bmatrix} c(\theta_4) & -s(\theta_4) & 0 & a_3 \\ s(\theta_4)c(\alpha_3) & c(\theta_4)c(\alpha_3) & -s(\alpha_3) & -s(\alpha_3)d_4 \\ s(\theta_4)s(\alpha_3) & c(\theta_4)s(\alpha_3) & c(\alpha_3) & c(\alpha_3)d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c(\theta_4) & -s(\theta_4) & 0 & 85 \\ 0 & 0 & -1 & -300 \\ s(\theta_4) & c(\theta_4) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

(13)

$$T_5^4(\theta_5) = \begin{bmatrix} c(\theta_5) & -s(\theta_5) & 0 & a_4 \\ s(\theta_5)c(\alpha_4) & c(\theta_5)c(\alpha_4) & -s(\alpha_4) & -s(\alpha_4)d_5 \\ s(\theta_5)s(\alpha_4) & c(\theta_5)s(\alpha_4) & c(\alpha_4) & c(\alpha_4)d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c(\theta_5) & -s(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s(\theta_5) & -c(\theta_5) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (14)$$

$$T_6^5(\theta_6) = \begin{bmatrix} c(\theta_6) & -s(\theta_6) & 0 & a_5 \\ s(\theta_6)c(\alpha_5) & c(\theta_6)c(\alpha_5) & -s(\alpha_5) & -s(\alpha_5)d_6 \\ s(\theta_6)s(\alpha_5) & c(\theta_6)s(\alpha_5) & c(\alpha_5) & c(\alpha_5)d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c(\theta_6) & -s(\theta_6) & 0 & 0 \\ 0 & 0 & -1 & -90 \\ s(\theta_6) & c(\theta_6) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (15)$$

As mentioned in above section, the forward kinematics of the end-effector with respect to the base frame is determined by multiplying those six transformation matrices [30-32].

$$T_{base}^{end\text{-}effector} = T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_5^4 T_6^5 \quad (16)$$

The total transformation in terms of all of the joint angles/variables can be alternatively represented as

$$T_{base}^{end\text{-}effector} = T_6^0 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (17)$$

with,

$$n_x = \big((s_1s_4 - c_1s_{23}c_4)c_5 - c_1c_{23}s_5\big)c_6 + (c_1s_{23}s_4 + s_1c_4)s_6$$
$$n_y = \big((-s_1s_{23}c_4 - c_1s_4)c_5 - s_1c_{23}s_5\big)c_6 + (s_1s_{23}s_4 - c_1c_4)s_6$$
$$n_z = (c_{23}c_4c_5 - s_{23}s_5)c_6 - c_{23}s_4s_6$$
$$o_x = \big((-s_1s_4 + c_1s_{23}c_4)c_5 + c_1c_{23}s_5\big)s_6 + (c_1s_{23}s_4 + s_1c_4)c_6$$
$$o_y = \big((s_1s_{23}c_4 + c_1s_4)c_5 + s_1c_{23}s_5\big)s_6 + (s_1s_{23}s_4 - c_1c_4)c_6$$
$$o_z = (-c_{23}c_4c_5 + s_{23}s_5)s_6 - c_{23}s_4s_6$$
$$a_x = (-c_1s_{23}c_4 + s_1s_4)s_5 + c_1c_{23}c_5$$
$$a_y = (-s_1s_{23}c_4 - c_1s_4)s_5 + s_1c_{23}c_5$$
$$a_z = c_{23}c_4c_5 + s_{23}c_5$$
$$p_x = \big((-c_1s_{23}c_4 + s_1s_4)s_5 + c_1c_{23}c_5\big)d_6 + d_4c_1c_{23} - a_3c_1s_{23} - a_2c_1s_2 + a_4c_1$$
$$p_y = \big((-s_1s_{23}c_4 - c_1s_4)s_5 + s_1c_{23}c_5\big)d_6 + d_4s_1c_{23} - a_3c_1s_{23} - a_2c_1s_2 + a_1c_1$$
$$p_z = (c_{23}c_4s_5 + s_{23}c_5)d_6 + d_4s_{23} + a_3c_{23} + a_2c_2$$

where c_i and s_i are the short hands of $\cos(\theta_i)$ and $\sin(\theta_i)$, respectively and c_{ij}, and s_{ij} are the short hands for $\cos\big(\theta_i + \theta_j\big)$ and $\sin\big(\theta_i + \theta_j\big)$, respectively. Then the position of the end effector in base/Cartesian coordinates can be represented as:

$$P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \quad (18)$$

and the corresponding Euler angles can be obtained using equations 19, 20, and 21.

$$\phi_z = atan\left(\frac{n_y}{n_x}\right) \quad (19)$$

$$\phi_y = atan\left(\frac{-n_z}{n_x cos(\phi_z) + n_y sin(\phi_z)}\right) \quad (20)$$

$$\phi_x = atan\left(\frac{a_x sin(\phi_z) - a_y cos(\phi_z)}{o_y cos(\phi_z) - o_x sin(\phi_z)}\right) \quad (21)$$

where atan is the arctangent function [35].

EXPERIMENTAL RESULTS: FORWARD KINEMATICS OF HP3 MANIPULATOR

A user-friendly MATLAB GUI executes the outlined FK. Users input DH values (Figure (6) for HP3), select angles/pulses, compute T, and extract Euler. Validation across five cases matches MotoSim EG: Home (Figures (7,8)), Second Home (Figures (9,10)), and three random (11-16). Identical outputs confirm model reliability.

Pressing the button "Export" will initialize all required parameters for the subsequent process of the interface. Next, the user can either specify the joint angles: $\theta_s, \theta_l, \theta_u, \theta_r, \theta_b$ and θ_t or the equivalent pulse counts: pulse $_s$, pulse $_l$, pulse $_u$, pulse $_r$, pulse $_b$ and pulse $_t$ using a radio button. Afterwards, by pressing the button "Forward", the user can obtain the total transformation matrix $T_{\text{base}}^{\text{end-effector}} = T_6^0$. The last column of the matrix represents the position of the end effector in base/Cartesian coordinates. Note that the GUI will automatically provide the equivalent joint angles and/or pulse counts depending on the type of the specified input arguments (joint angles or pulse counts). Finally, clicking on the button "Euler angles" yields the corresponding Euler angles. We have checked the functionality and the accuracy of our work/GUI for five different trials/tests and compared the results to that of Motoman Simulation Program (MotoSim EG) outputs. The first trial corresponds to the Home Position of the HP3 manipulator ($\theta_s = 0°, \theta_l = 90°, \theta_u = 0°, \theta_r = 0°, \theta_b = 0°$ and $\theta_t = 0°$ or equivalently pulse $_s = 0$, pulse $_l = 0$, pulse $_u = 0$, pulse $_r = 0$, pulse $_b = 0$ and pulse $_t = 0$). Figure (7) illustrates the position and orientation of the end effector in base/Cartesian coordinates and the corresponding pulse counts and joint angles for the Home Position of the HP3 manipulator using MotoSim. Figure (8) depicts the result of the MATLAB GUI for the same pulse counts or joint angles (Home Position). The position and orientation of the end-effector in base/Cartesian coordinates shown in both figures are indeed one and the same; which in turn prove the correctness of the proposed mathematical model for the forward kinematics of the manipulator at hand.

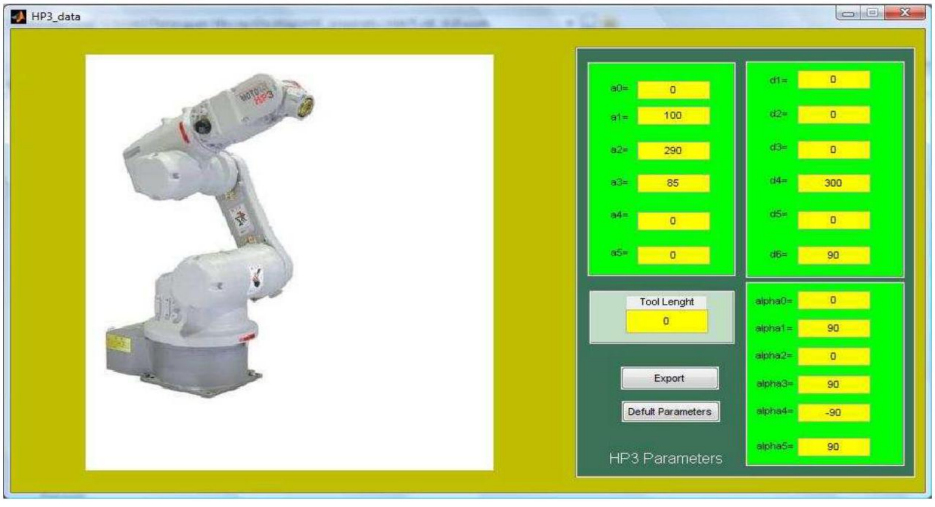

Figure 6: MATLAB GUI: specification of DH parameters for the HP3 manipulator.

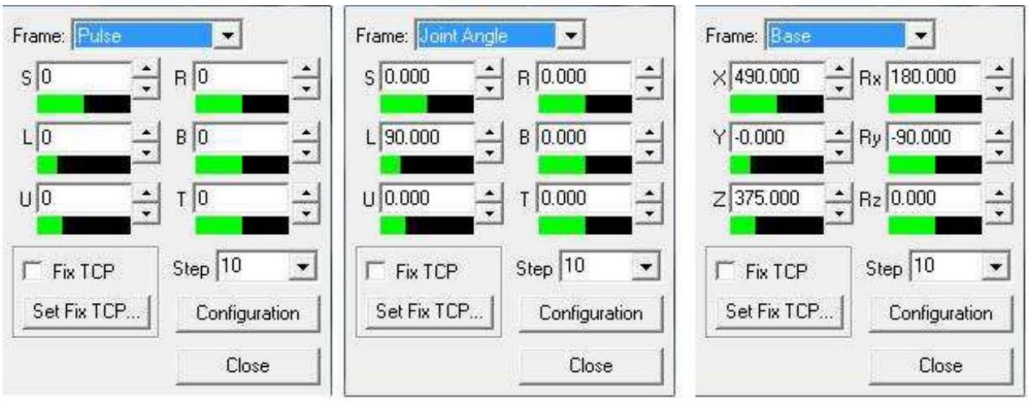

Figure 7: MotoSim EG: HP3's manipulator home position (Test 1).

Figure 8: MATLAB GUI: HP3's manipulator home position (Test 1).

Similarly, Figure (9) and Figure (10) show comparison of the results for the Second Home Position of the HP3 manipulator ($\theta_s = 0°, \theta_l = 90°, \theta_u = 0°, \theta_r = 0°, \theta_b = -90°$ and $\theta_t = 0°$ or equivalently pulse $_s = 0$, pulse $_l = 0$, pulse $_u = 0$, pulse $_r = 0$, pulse $_b = -81900$ and pulse $_t = 0$). The last three tests/comparisons (Figure (11) and Figure (12), Figure (13) and Figure (14), Figure (15) and Figure (16)) show the forward kinematics results for arbitrary joint angles and/or pulse counts. Based on all of the experimental results presented in this sub-section, one can rest assured that the functionality of the proposed forward kinematics process/algorithm/GUI for the HP3 manipulator is indeed correct.

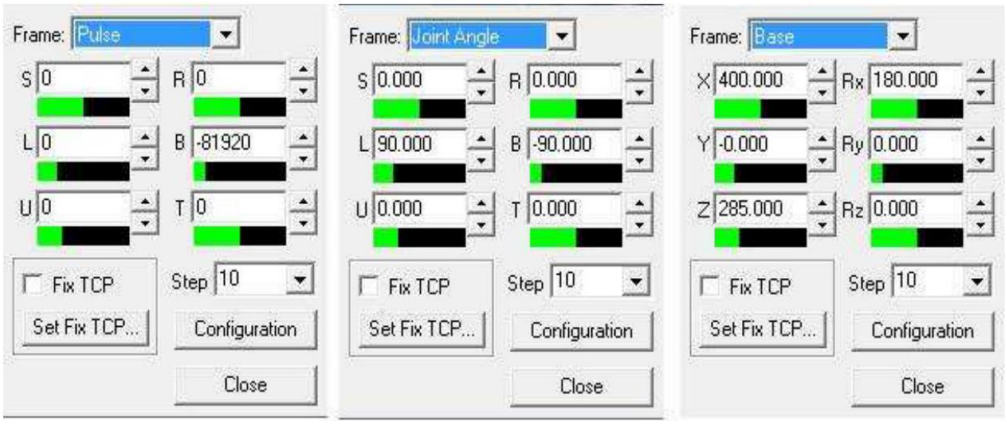

Figure 9: MotoSim EG: HP3's manipulator Second Home Position (Test 2).

Figure 10: MATLAB GUI: HP3's manipulator Second Home Position (Test 2).

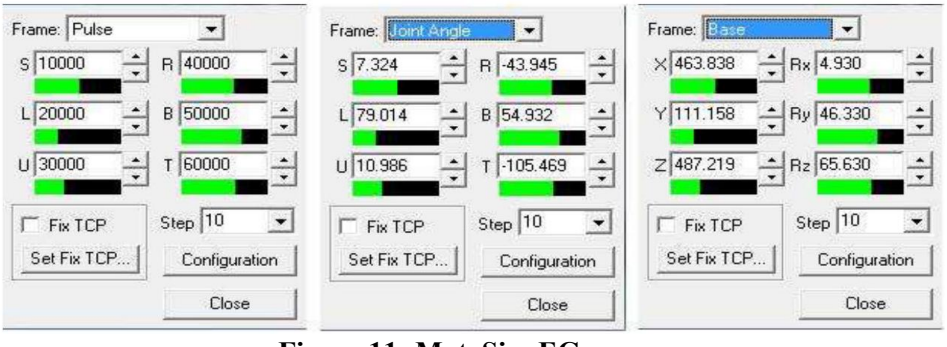

Figure 11: MotoSim EG.

Figure 12: MATLAB GUI: Test 3.

Figure 13: MotoSim EG: Test 4.

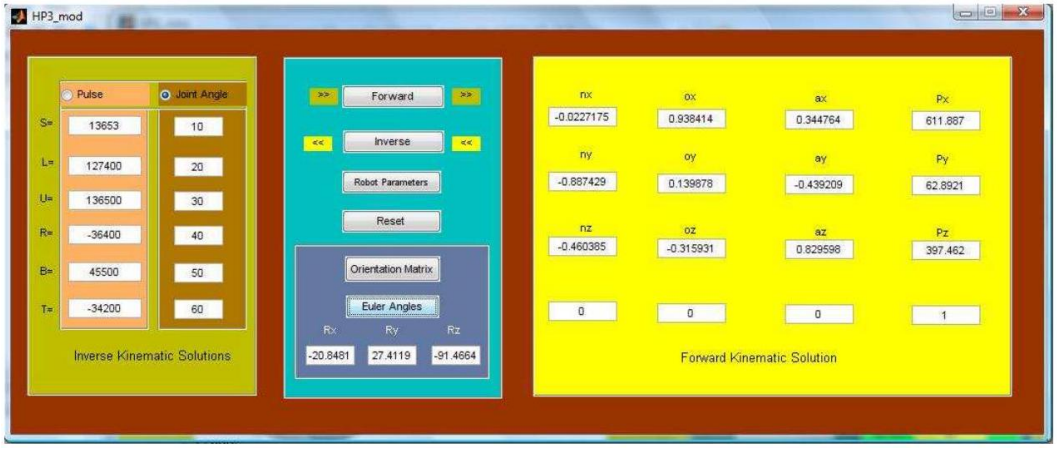

Figure 14: MATLAB GUI: Test 4.

Figure 15: MotoSim EG: Test 5.

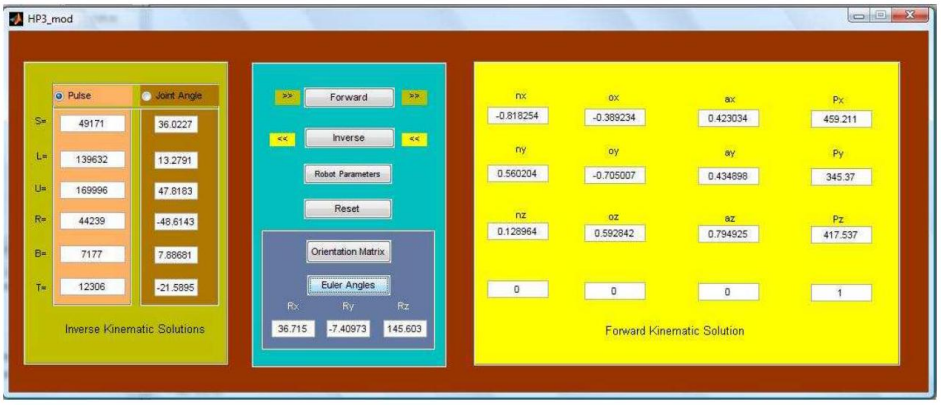

Figure 16: MATLAB GUI: Test 5.

Inverse Kinematics

IK computes joint values from end-pose, vital for planning and control, yet complicated by interdependencies, multiplicities, and singularities. Structure dictates solvability. Here, we blend analytical (geometric for θ1) and numerical (RBF-ANN for θ2/θ3) with analytical for θ4-θ6. n this paper, a hybrid approach combining two primary methods is proposed to solve the inverse kinematics problem of the HP3 manipulator. The analytical method determines the joint variables through analytical and geometric relationships based on the given configuration data, while the numerical method employs function approximation using a Radial Basis Function (RBF) Artificial Neural Network (ANN)).

Inverse Kinematics: Solving for θ Using Geometric and Analytic Methods

A simple strategy can be used to solve the inverse kinematics of the first joint angles (θ_s) by first deriving the position of the wrist (depicted in Figure (4) as P_w). Let the position and orientation of the end effector in Base/Cartesian coordinates be given as $P = [p_x, p_y, p_z, \phi_x, \phi_y, \phi_z]^T$ (as depicted in Figure (4)). The corresponding orientation matrix R_6^0 for the given Euler angles ϕ_x, ϕ_y and ϕ_z can be shown as follows [31].

$$R_6^0 = Euler(\phi_x, \phi_y, \phi_z) = \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = [n \quad o \quad a] \tag{22}$$

$$R_6^0 = \begin{bmatrix} c(\phi_y)c(\phi_z) & s(\phi_x)s(\phi_y)c(\phi_z) - c(\phi_x)s(\phi_z) & c(\phi_x)s(\phi_y)c(\phi_z) + s(\phi_x)s(\phi_z) \\ c(\phi_y)s(\phi_z) & s(\phi_x)s(\phi_y)s(\phi_z) + c(\phi_x)c(\phi_z) & c(\phi_x)s(\phi_y)s(\phi_z) - s(\phi_x)c(\phi_z) \\ -s(\phi_y) & s(\phi_x)c(\phi_y) & c(\phi_x)c(\phi_y) \end{bmatrix} \quad (23)$$

where s and c are the sine and cosine functions respectively. The wrist position (P_w) can be found using [33].

$$P_w = P - d_6 a \quad (24)$$

It is now possible to find the inverse kinematics for θ_s. The first joint angle can be found using [33,34].

$$\theta_s = atan\left(\frac{P_{wy}}{P_{wx}}\right) \quad (25)$$

The computation of the inverse kinematics for θ_l and θ_u is both mathematically complex and computationally demanding. This is primarily due to the dependency between the 2^{nd} and 3^{rd} joint angles as warned in previews section (MOTOMAN's axis representation) and due to singularities and nonlinearities. Hence, we propose a RBF-ANN neural network to solve the inverse kinematics for θ_l and θ_u [35].

Inverse Kinematics: Solving for θ_l and θ_u Using RBF Artificial Neural Network

To solve the inverse kinematics for θ_l and θ_u of the MOTOMAN HP3 manipulator, we employ a Radial Basis Function Artificial Neural Network (RBF-ANN). The RBF-ANN [36] consists of multiple layers: an input layer that forwards signals to the hidden layer, a hidden layer (or basis function layer) that typically uses functions such as the Gaussian function, and an output layer, which is usually a simple linear function. The RBF-ANN excels in local approximation, meaning that when input signals fall near the center of a basis function, the hidden layer produces a significant output [35, 36].

Using an ANN to study the inverse kinematics presents two main challenges: selecting an appropriate ANN type and generating a suitable training dataset. Considering calculation accuracy and training time, we adopt the network configuration illustrated in Figure (17), with a Gaussian function as the basis function. To achieve a closed-form solution and align with the joint ranges and desired workspace similar to [35], each joint angle range is defined as follows:

$$-45^\circ < \theta_s, \theta_u, \theta_r, \theta_b, \theta_t < 45^\circ \quad (26)$$

$$50^\circ < \theta_l < 125^\circ \quad (27)$$

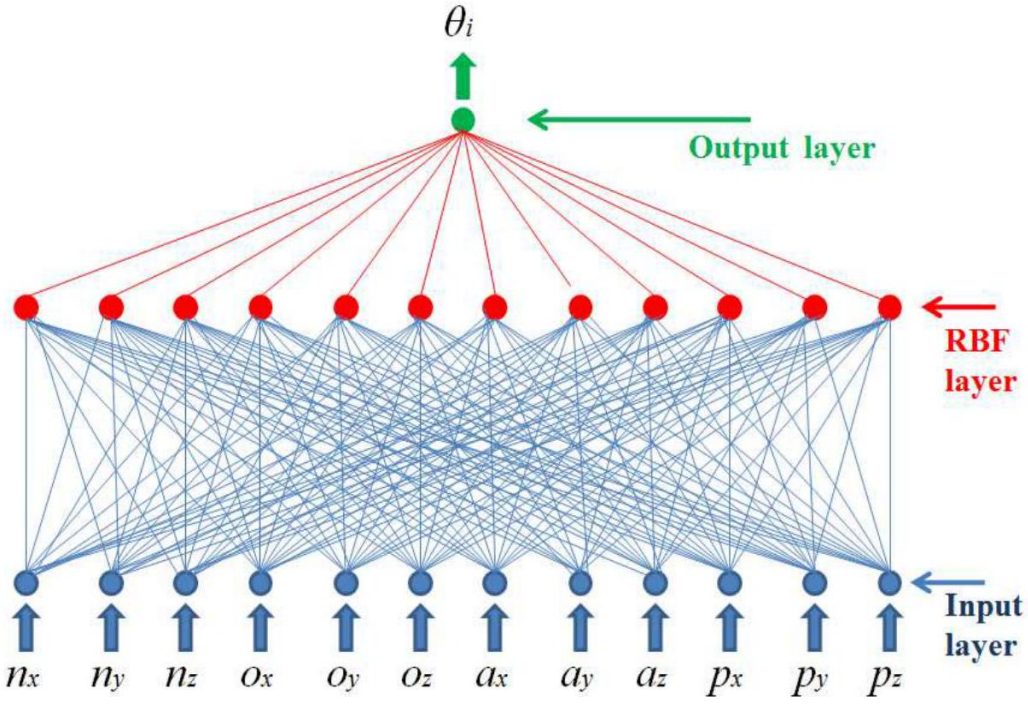

Figure 17: The structure of the RBF-ANN networks.

We use the previously derived forward kinematics to iteratively calculate the end-effector's pose, $T = [n_x, n_y, n_z, o_x, o_y, o_z, a_x, a_y, a_z, p_x, p_y]^T$ corresponding to the given joint-space configuration, $Q = [\theta_l, \theta_u]^T$ (for the specified joint angle ranges). Thus, we get the training data set $T \rightarrow Q$. The pose and orientation T in homogeneous coordinates are used as the network's input and the joint angles θ_l and θ_u are used as the network's output. For the two joint angles, we have constructed two networks to compute each joint angle (θ_l and θ_u). Note that, certainly, the 6 elements of P vectors can also be used as the networks input instead of the 12 elements of the T vectors.

We generated 7,776 training vectors (T) for various random joint angles (within the specified range) and trained two networks, for which each output corresponds to a joint angle (θ_l or θ_u). We have also generated a separate testing data (T vectors and their corresponding Q vectors) that is comprised of 216 vectors. After training the two networks using the training data, we tested their ability to generalize using the testing data. The Mean Square Errors (ΔE) were found not to exceed 0.0157 and 0.0242 for θ_l and θ_u respectively.

We believe that it is instructive to note from other studies by Karlik and Aydin [37] investigated the inverse kinematics solution of a 6-DOF manipulator using ANNs and concluded that a Back Propagation (BP) network with two hidden layers and a single output neuron performs better than a network with one hidden layer and six output neurons. Additionally, Zhang [35] demonstrated that RBF-ANN networks provide higher computational accuracy than BP networks and exhibit faster convergence rates.

Inverse Kinematics: Solving for θ_r, θ_b and θ_t Using Geometric and Analytic Methods

After successfully solving for the first three joint angles, we can now solve for the remaining three joint angles (θ_r, θ_b and θ_t). To determine the necessary joint angles

θ_r, θ_b and θ_t that correspond to the desired position and orientation of the end-effector, we simply take advantage of the previously computed joint angles ($\theta_s, \theta_l, \theta_u$) and the special configuration of the last three joints. Because the orientation of the end-effector is defined by R_6^0, it's simple to get R_6^3. However, before finding R_6^3, one needs to convert the computed joint angles as discussed (due to MOTOMAN's axis representation). In other words;

$$\theta_1 = \theta_s \tag{28}$$

$$\theta_2 = \theta_l - 90^\circ \tag{29}$$

$$\theta_3 = \theta_u - \theta_l + 90^\circ \tag{30}$$

Then R_6^3 can be easily computed as follows:

$$R_6^3 = (R_5^0)^{-1} R_6^0 = (R_5^0)^T R_6^0 \tag{31}$$

$$R_6^3 = \begin{bmatrix} -c_1 s_{23} & -s_1 s_{23} & c_{23} \\ -c_1 c_{23} & -s_1 c_{23} & -s_{23} \\ s_1 & -c_1 & 0 \end{bmatrix} \times \begin{bmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{32}$$

with

$r_{11} = -c_1 s_{23} n_x - s_1 s_{23} n_y + c_{23} n_z$, $r_{12} = -c_1 s_{23} o_x - s_1 s_{23} o_y + c_{23} o_z$

$r_{13} = -c_1 s_{23} a_x - s_1 s_{23} a_y + c_{23} a_z$, $r_{21} = -c_1 c_{23} n_x - s_1 c_{23} n_y - s_{23} n_z$

$r_{22} = -c_1 c_{23} o_x - s_1 c_{23} o_y - s_{23} o_z$, $r_{23} = -c_1 c_{23} a_x - s_1 c_{23} a_y - s_{23} a_z$

$r_{31} = s_1 n_x - c_1 n_y$, $r_{32} = s_1 o_x - c_1 o_y$, $r_{33} = s_1 a_x - c_1 a_y$

where s_{23} and c_{23} are $\sin(\theta_2 + \theta_3)$ and $\cos(\theta_2 + \theta_3)$ respectively. It is now possible to obtain the solution for θ_r, θ_b and θ_t [33].

For $\theta_b \in [0, \pi]$ the solution is

$$\theta_r = \operatorname{atan}\left(\frac{r_{33}}{r_{13}}\right) \quad , \quad \theta_b = \operatorname{atan}\left(\frac{\sqrt{r_{13}^2 + r_{33}^2}}{-r_{23}}\right) \quad , \theta_t = \operatorname{atan}\left(\frac{-r_{22}}{r_{21}}\right)$$

For $\theta_b \in [-\pi, 0]$ the solution is

$$\theta_r = \operatorname{atan}\left(\frac{-r_{33}}{-r_{13}}\right) , \theta_b = \operatorname{atan}\left(\frac{\sqrt{r_{13}^2 + r_{33}^2}}{r_{23}}\right) , \theta_t = \operatorname{atan}\left(\frac{r_{22}}{-r_{21}}\right)$$

This concludes the process of solving the IK problem for the HP3 manipulator.

EXPERIMENTAL RESULTS: INVERSE KINEMATICS OF HP3 MANIPULATOR

In this subsection, we present the accuracy of the inverse kinematics model and the solution derived in the previous subsection. Table (2) provides a summary of the performance of the proposed approach. The table shows the results for seven different and randomly acquired testing vectors (for the specified joint angle ranges but not seen by any of the two RBF-ANN networks). We have trained two different RBF-ANN neural networks for each joint angle (θ_l, θ_u). One operating with 300 neurons (RBF1) and the other operating with 650 neurons (RBF2). It is a known fact that the generalizability of

an RBF-ANN network depends not only on the training data but also on the specified spread (σ) and on the specified number of neurons [38-41].

A cross-validation analysis-trained on the training dataset and evaluated on the testing dataset-was conducted across various logarithmically spaced spread values to identify the optimal spread., $\sigma = 650$. It should be noted that increasing the spread results in a smoother function approximation. However, if the spread is too large, a greater number of neurons will be needed to accurately model a rapidly changing function [42–44]. Conversely, a spread that is too small requires many neurons to fit even a smooth function, which can negatively affect the network's ability to generalize. The effect of varying the number of neurons is illustrated in Table (4). It portrays that the approximation for the joint angles (θ_l, θ_u) varies accordingly. RBF1 seems to be generalizing very well for small joint angle ranges and RBF2 appears to be the favored network for wider joint angle ranges. However, let it be known that we have observed the contrary for other examples/tests. All in all, based on these experimental results, one can safely presume that a plausible solution for the inverse kinematics of the MOTOMAN HP3 manipulator (for specific joint angles ranges) is indeed obtained. Note that such solution has been implemented in the MATLAB GUI as well [45-47].

Table 4: Inverse Kinematics solution experimental results for HP3 manipulator. RBF1 utilizes 300 neurons and RBF2 utilizes 650 neurons.

No		θ_s	θ_l	θ_u	θ_r	θ_b	θ_t	ΔE
1	Desired Value	−10.23°	114.9°	32.26°	16.05°	−6.83°	1.03°	
	IK (RBF1)	−10.23°	114.88°	32.28°	16.01°	−6.84°	1.07°	0.0212
	IK (RBF2)	−10.23°	114.6°	32.45°	15.64°	−7.0°	1.44°	0.2249
2	Desired Value	27.01°	110.8°	24.19°	4.44°	−21.39°	14.02°	
	IK (RBF1)	27.01°	110.72°	24.22°	4.43°	−21.42°	14.03°	0.0230
	IK (RBF2)	27.01°	110.9°	24.23°	4.43°	−21.43°	14.03°	0.0311
3	Desired Value	−30.05°	109.52°	8.25°	−15.46°	−21.79°	23.22°	
	IK (RBF1)	−30.05°	109.58°	8.24°	−15.47°	−21.78°	23.22°	0.0158
	IK (RBF2)	−30.05°	109.82°	8.11°	−15.55°	−21.66°	23.32°	0.1278
4	Desired Value	−16.04°	65.93°	−5.85°	−8.64°	20.27°	2.29°	
	IK (RBF1)	−16.03°	65.89°	−5.85°	−8.64°	20.27°	2.29°	0.008
	IK (RBF2)	−16.03°	65.88°	−5.95°	−8.64°	10.36°	2.25°	0.054
5	Desired Value	41.84°	115.59°	−34.92°	−22.28°	−21.13°	−24.47°	
	IK (RBF1)	41.83°	115.63°	−33.82°	−21.2°	−21.13°	−24.47°	0.733
	IK (RBF2)	41.79°	115.01°	−33.98°	−22.03°	−20.87°	−24.12°	0.659
6	Desired Value	−14°	57°	11.88°	−4.21°	−43.91°	8.96°	
	IK (RBF1)	−14°	57.61°	11.51°	−4.24°	−43.54°	8.99°	0.236
	IK (RBF2)	−14°	56.89°	12.02°	−4.2°	−44.01°	8.95°	0.229
7	Desired Value	27.85°	101.57°	7.66°	14.51°	22.92°	22.88°	
	IK (RBF1)	27.85°	101.39°	7.92°	14.67°	22.68°	22.72°	0.166
	IK (RBF2)	27.85°	101.67°	7.69°	14.53°	22.88°	22.85°	0.0366

The following subsections detail the network architecture, data generation, training procedure, and parameter selection to ensure reproducibility [48-50].

1. Network Input Vector Selection

After preliminary testing, the input to the RBF-ANN was chosen as a 6-element pose vector derived from the position and orientation of the end-effector, defined as: Input = $[p_x, p_y, p_z, \varphi_x, \varphi_y, \varphi_z]^T$, where $[p_x, p_y, p_z]$ is the end-effector position and $[\varphi_x, \varphi_y, \varphi_z]$ are the Z-Y-X Euler angles extracted from the rotation matrix of the homogeneous transformation T_0^6. This representation was selected over the full 12-element T-matrix vector to reduce redundancy and computational load, without a significant loss in accuracy for the specified workspace as shown in Figure (17).

2. Training and Testing Data Generation

A comprehensive dataset was generated using the forward kinematics model derived in previews section The joint angles θl and θu were varied within their specified ranges ($50° < \theta l < 125°$ and $-45° < \theta u < 45°$), while the other joints (θ_s, θ_r, θ_b, θ_t) were randomly sampled from a uniform distribution within their ±45° range.

To ensure even coverage of the joint space, a stratified grid sampling approach was used for θl and θu. The range of θl was divided into 15 intervals and θu into 12 intervals, creating a 15x12 grid. From each grid cell, 45 unique random samples of the other four joints were drawn, resulting in a total of 15 * 12 * 45 = 8,100 data points. After removing configurations leading to self-collisions or exceeding the physical workspace, the final training set consisted of 7,776 input-output pairs. A separate, entirely disjoint testing set of 216 vectors was generated using a different random seed to evaluate the network's generalization performance.

3. RBF-ANN Architecture and Training Algorithm

A separate RBF-ANN was constructed for each joint angle (θ_l and θ_u). Each network had the following structure:

- Input Layer: 6 neurons (for the 6-element pose vector)
- Hidden Layer: Comprised M RBF neurons. We investigated two configurations: RBF1 with M=300 neurons and RBF2 with M=650 neurons
- Output Layer: 1 linear neuron (providing the estimated joint angle)

The training process involved the following steps for each network:

- Center Selection (c_i): The centers of the RBF neurons were established by applying the K-means clustering algorithm to the 7,776 training input vectors. This ensures the centers are representative of the data distribution in the input space.
- Spread Calculation (σ_i): A common spread parameter σ was used for all neurons in a given network. The value was determined via k-fold cross-validation (with k=5) on the training set. The optimal value was found to be σ = 650, which produced the smoothest function approximation without overfitting.
- Output Weight Calculation (w_i): The output weights were computed using the pseudo-inverse (least-squares) method. The hidden layer activation matrix H was constructed, where each element H_{ij} is the output of the j-th Gaussian kernel for the i-th training sample. The output weights w was then calculated as $w = (H^T H)^{-1} H^T y$, where y is the vector of target joint angles from the training data.

CONCLUSION

From a mathematical and set-theoretic perspective, the relationship between forward kinematics (FK) and inverse kinematics (IK) can be viewed as a nonlinear mapping between the robot manipulator's joint space and its operational (Cartesian) space. This paper presents both the forward kinematics and a feasible solution for the inverse kinematics problem of the MOTOMAN HP3 manipulator. Forward kinematics, which maps joint space to Cartesian space in a one-to-one manner, is generally considered a straightforward task. The paper shows that by using the DH algorithm, it was possible to successfully obtain the exact transformation function. Unlike the FK problem, the inverse kinematics problem is a tricky one. This paper presents a hybrid method comprised of analytic and numerical approximation to solve the IK problem. This paper applies analytical/geometrical methods and two RBF-ANN networks that consist of twelve input neurons and one output neuron to solve the IK problem of the MOTOMAN manipulator. Considering the powerful ability of an Artificial Neural Network to process nonlinear mapping relations, we believe that it is extremely useful in approximating IK solutions for most complex industrial or commercial manipulators. For example, the DA10 manipulator does present 15 joints and hence an IK solution by an algebraic method is almost impossible. However, certain types of artificial neural networks (ANNs) are ideally suited for this problem, as they eliminate the need for the traditional, complex process of deriving equations and programming. Although the new/proposed IK solution appears promising, further tuning/optimization of the different RBF-ANN parameters (number of neurons and spread) could improve the overall accuracy of the results. A valuable area of future work would be to approximate a closed-form IK solution for a wider range of joint angles (instead of the ranges specified in this paper). In doing so, one is advised to augment the number of training vectors. Furthermore, different types of ANNs—such as Feedforward Backpropagation Networks (FFBN), Trainable Cascade-Forward Backpropagation Networks (TCFBN), Generalized Regression Neural Networks (GRNN), or Probabilistic Neural Networks (PNN) can be employed depending on the specific inverse kinematics problem being addressed.

FUTURE WORK

- **Integration with Trajectory Planning and Control:** The kinematics model is the foundation for motion
    - Combine the kinematic model with the manipulator's dynamics to develop a dynamic control scheme (e.g., Computed Torque Control) for more accurate trajectory tracking under load
- **Calibration and Accuracy Improvement:** Move from a theoretical model to a physically accurate one
    - Perform kinematic calibration to identify the actual DH parameters of a physical MOTOMAN HP3 robot, compensating for manufacturing imperfections

REFERENCES

[1] Fouz, M. and Rezeka, S. (2013). Neural-networks-based inverse kinematics for a robotic manipulator, *15th International Conference on Aerospace Sciences & Aviation Technology*, ASAT, 1-18. doi:10.21608/ASAT.2013.22084.

[2] Ari, M. and Mondada, F. (2018). *Kinematics of a robotic manipulator*, in Book Chapter, doi: 10.1007/978-3-319-62533-1_16.

[3] Jha, P. (2015). Inverse kinematic analysis of robot manipulators, PhD thesis, National Institute of Technology Rourkela, India.
[4] Sharkawy, A. and Khairullah, S. (2023). Forward and inverse kinematics solution of a 3-DOF articulated robotic manipulator using artificial neural network, *International Journal of Robotics and Control Systems*, 3(2). doi: 10.31763/ijrcs.v3i2.1017.
[5] Sharkawy, A. (2022). Forward and inverse kinematics solution of a robotic manipulator using a multilayer feedforward neural network, *Journal of Mechanical and Energy Engineering*, 6(2). doi: 10.30464/jmee.00300.
[6] Virgala, I. and Prada, E. (2020). *Kinematics of serial manipulators*, in Book Chapter, Jul. 2020. doi: 10.5772/INTECHOPEN.93138.
[7] Abougarair, A., Aburakhis, M., & Edardar, M. (2022). Adaptive neural networks based robust output feedback controllers for nonlinear systems, *International Journal of Robotics and Control Systems*, 2(1), 37-56. doi: https://doi.org/10.31763/ijrcs.v2i1.523.
[8] Hroncová, D., Miková, L., Prada, E., Rákay, R., Ján Sincák P. and Merva, T. (2022). Forward and inverse robot model kinematics and trajectory planning, *20th International Conference on Mechatronics - Mechatronika*, Pilsen, Czech Republic, 1-9. doi: 10.1109/ME54704.2022.9983355.
[9] Yoshimitsu, Y. and Ikemoto, S. (2023). Forward/inverse kinematics modeling for tensegrity manipulator based on goal-conditioned variational autoencoder, *in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. doi: 10.1109/IROS55552.2023.10341525.
[10] Buzkhar, I. and et al. (2023). Modeling and Control of a Two-Wheeled Robot Machine with a Handling Mechanism, *2023 IEEE 3rd International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023)*, Libya. doi. 10.1109/MI-STA57575.2023.10169424.
[11] Antonov, A. and Glazunov, V. (2021). Inverse and forward kinematics and workspace analysis of a novel 5-DOF (3T2R) parallel-serial (hybrid) manipulator, *International Journal of Advanced Robotic Systems*, 18(2). doi: 10.1177/1729881421992963.
[12] Enev, S. (2024). Direct and inverse kinematics solutions of a particular type of 3-PSP parallel manipulator, *IOP Conference Series: Materials Science and Engineering*, 1317(1). doi: 10.1088/1757-899X/1317/1/012007.
[13] Saleem, M. and Khan, A. (2024). Analytical kinematics framework for the control of a parallel manipulator - A generalized kinematics framework for parallel manipulators, *In Proceedings of the 6th International Conference on Informatics in Control, Automation and Robotics,* 1: ICINCO, ISBN 978-989-674-000-9, pages 280-286. doi: 10.5220/0002214102800286.
[14] Samartín J. and Barrientos, A. (2023). Kinematic modelling of a 3RRR planar parallel robot using genetic algorithms and neural networks, *Machines*, 11(10). doi: 10.3390/machines11100952.
[15] Buzkhar, I. and et. al. (2023). Design and Implementation of Hydric Controller for Two Wheeled Robot, *IJEIT on Engineering and Information Technology*, 11(1).doi: https://doi.org/10.36602/ijeit.v11i1.5
[16] Gnan, H. and et. al. (2022). Real Time Classification for Robotic Arm Control Based Electromyographic Signal, *2022 IEEE 2st International Maghreb Meeting*

of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2022), Sabrata, Libya. doi: 10.1109/MI-STA54861.2022.9837703

[17] Desai, R. and Muthuswamy, S. (2021). A forward, inverse kinematics and workspace analysis of 3RPS and 3RPS-R parallel manipulators, *Iranian Journal of Science and Technology*, Transactions of Mechanical Engineering, 45(2). doi: 10.1007/S40997- 020-00346-9.

[18] Attawil, I. and et. al. (2024). Enhancing Lateral Control of Autonomous Vehicles through Adaptive Model Predictive Control, *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA)*, Tripoli, Libya. doi: 10.1109/MI-STA61267.2024.10599733

[19] Crane, C. and Duffy, J. (2009). Kinematic analysis of robot manipulators, (Book).

[20] Agustian, I. and Faurina, R. (2021). Robot manipulator control with inverse kinematics PD- pseudoinverse Jacobian and forward kinematics Denavit Hartenberg, *Journal of Engineering and Technology*, 21(1). doi: 10.14203/JET.V21.8-18.

[21] Talli, A. and Marebal, D. (2021). Kinematic analysis and simulation of robotic manipulator based on RoboAnalyzer, in Book Chapter. doi: 10.1007/978-981-16-0336-5_6.

[22] Antonov, A. (2023), Inverse kinematics of a 5-DOF hybrid manipulator, *Automation and Remote Control*, 84(3). doi: 10.1134/S0005117923030037.

[23] Lan, L. and Yang, W. (2020). Learning the kinematics of a manipulator based on VQTAM, *Symmetry*, 12(4). doi: 10.3390/SYM12040519.

[24] Tripathi, A. and Gopaliya, S. (2023). Study of kinematics for industrial robots, in Book Chapter. doi: 10.1007/978-981-99-2349-6_23.

[25] Singh A. and Singla, A. (2016). Kinematic modeling of robotic manipulators, Proceedings of the National Academy of Sciences, *India – Section*, 87(3) , 303-319. doi: 10.1007/S40010-016-0285-X.

[26] Abougarair, A. (2023). Adaptive Neural Networks Based Optimal Control for Nonlinear System , *2023 IEEE 3rd International Maghreb Meeting Stabilizing of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA2023)*, Libya. doi: 10.1109/MI-STA57575.2023.10169340.

[27] Elmolihi, A., et al. (2020). Robust Control and Optimized Parallel Control Double Loop Design for Mobile Robot, *IAES International Journal of Robotics and Automation (IJRA)*, 9(3). doi: http://doi.org/10.11591/ijra.v9i3.pp160-170.

[28] Li, J. and Wang, Z. (2024). Transformer-based inverse kinematics for complex manipulators, *IEEE Robotics and Automation Letters*, 9(2), pp. 1234–1241.

[29] Chen, X. and Zhang, Y. (2023). Geometric inverse kinematics with deep learning for roboticmanipulators, *Robotics and Autonomous Systems*, 159, 104289.

[30] Craig, Introduction to Robotics: Mechanics and Control, Reading, MA: Addison-Wesley, 1990.

[31] Paul, R. (1982). Robot Manipulators: Mathematics, Programming, and Control. Cambridge, MA: MIT Press.

[32] Cubero, S. (2007). Industrial Robotics: Theory, Modelling and Control. Germany: Pro Literatur Verlag.

[33] Pires, N. (2006). Robot manipulators: Kinematic analysis and control, *Industrial Robot*, 33(4), 296–302. doi: 10.1007/978-0-387-23326-0_2.

[34] Niku, S. (2001). Introduction to Robotics: Analysis, Systems, Applications. Upper SaddleRiver, NJ: Prentice Hall.
[35] Dinh, B. and Dunnigan, M. (2014). A radial basis function network approach to approximate the inverse kinematics of a robotic system, *International Journal of Modelling Identification and Control*, 21(2), 113-124. doi: 10.1504/IJMIC.2014.060005.
[36] Wen, J. and Kreutz-Delgado, K. (2000). Neural network control of robot manipulators, *IEEE Transactions on Neural Networks*, 11(3), 567–579.
[37] Karlik, B. and Aydin, S. (2000). An improved approach to the solution of inverse kinematics prob- lems for robot manipulators, *Engineering Applications of Artificial Intelligence*, 13(2), 159-164. doi:10.1016/S0952-1976(99)00050-0.
[38] Abougarair, A. (2018). Virtual Reality Animation of ANFIS Controller for Mobile Robot Stabilization, *Journal of Engineering Research*, 25. https://jer.ly/PDF/Vol-25-2018/JER-08-25.pdf.
[39] Abougarair, A. (2019). Model Reference Adaptive Control and Fuzzy Optimal Controller for Mobile Robot, *Journal of Multidisciplinary Engineering Science and Technology*, 6(3), 9722-9728. Germany. https://www.jmest.org/wp-content/uploads/JMESTN42352870.pdf.
[40] Abougarair, A. (2020). Neural Networks Identification and Control of Mobile Robot Using Adaptive Neuro Fuzzy Inference System, *ICEMIS'20: Proceedings of the 6th International Conference on Engineering & MIS 2020*, https://doi.org/10.1145/3410352.3410734.
[41] Gnan, H. et al. (2021). Implementation of a Brain-Computer Interface for Robotic Arm Control, *(MI-STA2021),* Tripoli, Libya. doi: 10.1109/MI-STA52233.2021.9464359.
[42] Edardar, M. et al. (2021). Tracking Control with Hysteresis Compensation Using Neural Networks, *(MI-STA2021)*, Libya. doi: 10.1109/MI-STA52233.2021.9464365.
[43] Aburakhis, M. et al. (2022). Performance of Anti-Lock Braking Systems Based on Adaptive and Intelligent Control Methodologies, *Indonesian Journal of Electrical Engineering and Informatics (IJEE)*. doi: 10.52549/ijeei.v10i3.3794.
[44] Ellafi, M. et al. (2023). Analysis of Mobile Accelerometer Sensor Movement Using Machine Learning Algorithms, *(MI-STA2023)*. doi: 10.1109/MI-STA57575.2023.10169214
[45] Ma'arif, A. et al. (2024). Model Predictive Control for Optimizes Battery Charging Process, *2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering*. doi: 10.1109/MI-STA61267.2024.10599662.
[46] Bakouri, M. et al. (2024). Optimizing cancer treatment using optimal control theory, *AIMS Mathematics*, 9(11), 31740-31769. doi: 10.3934/math.20241526.
[47] Abougarair, A. (2022). Position and Orientation Control of a Mobile Robot Using Intelligent Algorithms Based Hybrid Control Strategies, *Journal of Engineering Research* (Libya), 34, 67-86. https://jer.ly/PDF/Vol-34-2022/JER-05-34-Abstract.php?f=a.
[48] Alaktiwi, A. et al. (2025). Adaptive Control Approach for Optimized Lane Keeping in Autonomous Vehicles, 4(1). doi.org/10.51984/sucp.v4i1.3956.

[49] Zrigan, A. (2025). Integration of MPC and SOLADRC to Optimize PWR Performance, *Journal of Pure and Applied Sciences (JOPAS)*, 4(1). doi: 10.51984/SUCP.V4I1.3870.
[50] Guma, W. et al. (2024). Implementation and Performance Evaluation of Intelligent Techniques for Controlling a Pressurized Water Reactor, *Journal of Automation, Mobile Robotics and Intelligent Systems*, 4, 71-85. doi.org/10.14313/JAMRIS/4-2024/32.