



## Research Article

# Modeling and Simulation Validation of a Lightweight Indigenous Six-Degrees-of-Freedom Robotic Manipulator (Dexter-6Pro)

Kinza Fatima<sup>1</sup> , Hajra Qureshi<sup>2</sup> , Muhammad Ahsan<sup>3</sup> , Muhammad Rzi Abbas<sup>4</sup> and Zaryab Basharat<sup>5\*</sup>

<sup>1,2,3,4</sup>Department of Mechatronics and Control Engineering, University of Engineering and Technology (UET), Lahore, Pakistan

<sup>5</sup>MOE Key Laboratory of Thermo-Fluid Science and Engineering, Xi'an Jiaotong University, Xi'an, China

## Article Information

## Abstract

## Article History

Received: 20 August 2026

Revised: 18 September 2026

Accepted: 30 September 2026

Published online: 5 October 2026

## Keywords

Control of Robotic Systems

Dynamics

Manipulator Design and Kinematics

Position Control

Robot Dynamics

Torque Control

The growing demand for inexpensive, locally produced robotic manipulators in developing nations has necessitated the development of efficient modeling and control techniques for both industry and research. This paper presents the modeling and simulation of Dexter-6Pro, a 6-DOF lightweight robotic manipulator developed for such applications. The kinematics and dynamics models formulated using the Euler-Lagrange method, are verified in MATLAB and Simulink after being adapted from established research platforms, with adjustments to the Jacobian for dynamic computation. This study also presents the mechanical realization of Dexter-6Pro, a lightweight modular manipulator that is locally manufactured. A PD controller with feedforward torque compensation is used to validate the performance tracking. The precision and reliability of the symbolic model are analyzed with Simulink's built-in blocks of kinematics and dynamics. The simulation results show precise trajectory tracking and reliable performance throughout the workspace of the manipulator. To promote further study and reproducibility, all of the created MATLAB and Simulink materials for this research are publicly shared. The framework proposed sets up the basis for real-time deployment and supports the construction of local robotic solutions.

## Correspondence\*

2020me521@student.uet.edu.pk

## ORCID

Kinza Fatima

<https://orcid.org/0009-0007-3650-5667>

Hajra Qureshi

<https://orcid.org/0009-0000-3008-2332>

Muhammad Ahsan

<https://orcid.org/0000-0002-0221-0939>

Muhammad Rzi Abbas

<https://orcid.org/0000-0002-7748-0407>

Zaryab Basharat

<https://orcid.org/0009-0008-1914-3331>

© 2026 Centre for Research and Innovation (CRI). This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

## I. INTRODUCTION

Robotic manipulators with six joints (6-DOF) are essential to contemporary automation applications since they may assist with surgeries, precise assembly, welding, and inspection. In these domains, their exceptional dexterity, repeatability, and motion accuracy render them invaluable [1] [2]. The performance of these systems depends on the accuracy of their kinematic and dynamic models, which form the foundation for control design, trajectory planning and seamless simulation to hardware integration. Using off-the-shelf modular components to create lightweight robotic manipulators has become more popular in recent years as a

scalable and affordable way to quickly prototype ideas. Abbas *et al.* [3], for example, offered a development approach that prioritizes hardware realization using lightweight and modular design concepts. Accurate kinematic and dynamic models are necessary to enable dependable simulation, control design and practical implementation in order to support such physical properties. In some studies, the kinematic modeling, particularly forward kinematics, is implemented using MATLAB/Simulink [4] [5]. However, these implementations often lack thorough validation, especially regarding workspace evaluation, and Jacobian matrix consistency for dynamics. Similarly, symbolic dynamic modeling using the Euler-Lagrange

formulation is rare and, when present, frequently overlooks critical transformation errors related to center-of-mass positions and their effect on torque computations. In this work, we introduce a comprehensive modeling and simulation environment for the Dexter-6 Pro, a lightweight

6-DOF robotic manipulator created at UET Lahore's Human-Centered Robotics Lab developed using the framework proposed in [3]. The fabricated prototype of Dexter-6 Pro is shown in Figure 1, which validates the design geometry and assembly feasibility consistent with the simulated model.



Fig.1 Fabricated Dexter-6 Pro Prototype Developed at HCRL, UET Lahore

The contributions are as follows:

1. The Euler–Lagrange formulation is employed to obtain closed-form dynamic equations, such as the inertia, Coriolis, and gravity terms. A frame-consistent CoM transformation method is presented to make sure that the physically correct Jacobian and dynamic matrices.
2. The resulting model is simulated in MATLAB/Simulink and compared with Simscape Multibody results.
3. A locally manufactured lightweight manipulator, Dexter-6Pro, that validates the geometric and structural feasibility of the derived models.
4. The entire modeling and control codebase is published on a GitHub repository for open-access research and reproducibility.

By addressing some major modeling inconsistencies and providing a completely symbolic, simulation-validated formulation, this research enhances the precise control-oriented modeling of lightweight manipulators. It supports ongoing hardware development work and provides a solid foundation for future extension with sensor integration, real-time control, and experimental validation.

## II. RELATED WORK

Six-DOF robotic manipulators have emerged as an essential part of both industrial automation and scholarly research because of their ability to perform complex, multi-axis tasks accurately. [2] Comprehensive foundations for understanding the dynamics, kinematics, and control strategies used in robotic arms are provided by the foundational books by Siciliano *et al.* [6], Spong *et al.* [7]. More recent studies for example, Elhadidy *et al.* [1] provide contemporary insights into evaluating the accuracy and efficiency of kinematic modeling tools for 6-DOF industrial manipulators, particularly in practical case studies such as those involving KUKA robots. Numerous studies have shown modeling efforts tailored to 6-DOF manipulators. While Villalobos *et al.* [8] [9] concentrated on singularity analysis and inverse

kinematic solutions utilizing DH-based techniques, Kebria *et al.* [5] used MATLAB to create the kinematic and dynamic models of the UR5 manipulator. Sliding mode control was used on an IRB-120 robot by Barhaghtalab *et al.* [4], who investigated dynamic simulation and showed control robustness in the face of disturbances. In line with the approaches used in our project, Kufieta's work [10] and his open-source modeling scripts offer a thorough method for force calculation utilizing geometric Jacobians and Newton-Euler formulations. Abbas *et al.* [3] demonstrated the experimental development of lightweight manipulators built from off-the-shelf components, with an emphasis on reducing the design cycle time, in contrast to commercial systems.

Their research contributes to the continuous movement in robotics toward open, adaptable hardware solutions. Our manipulator, Dexter-6 Pro, builds upon this vision by offering a fully documented, low-cost robotic arm that supports bottom-up modeling, from analytical kinematics and dynamic simulations in Simulink to custom control implementations. Additionally, interdisciplinary compilations like “Advances in Automation and Robotics” Research [11], and proceedings from IEEE conferences [12], reflect the growing importance of integrating control theory, machine learning, and real-time systems into robotic manipulation research. These works also highlight the growing demand for open, adaptable platforms such as Dexter-6 Pro that can serve as a bridge between simulation environments and physical implementations. Although many studies address the modeling and control of 6-DOF manipulators, few provide a complete and transparent pipeline suitable for low cost, educational, or experimental platforms. This work addresses that gap by offering a fully documented framework that enables end-to-end control, validation, and system analysis. Table I provides a comparative overview of recent modeling studies and highlights how the proposed approach integrates kinematic modeling, dynamic formulation, and control verification into a unified workflow.

TABLE I COMPARISON OF RELATED 6-DOF MANIPULATOR MODELING AND CONTROL

| Recent Studies             | Inverse Kinematics          | Inverse Dynamics                                                                    | Control Simulation                                | Validation in MATLAB |
|----------------------------|-----------------------------|-------------------------------------------------------------------------------------|---------------------------------------------------|----------------------|
| Kufieta <i>et al.</i> [10] | Not addressed               | Addressed analytically; implementation and validation not reported                  | Not addressed                                     | Not reported         |
| Kebria <i>et al.</i> [5]   | Implemented at basic level  | Implemented using standard simulation blocks; no symbolic formulation               | Implemented (PD in Simulink Environment)          | Reported             |
| Abbas <i>et al.</i> [3]    | Not addressed               | Focused on actuator-level modeling                                                  | PI Implemented at actuator level (frameless BLDC) | Reported             |
| Present work               | Fully derived (closed form) | Symbolic modeling and analysis through implementation and validation through MATLAB | Implemented (CTC in Simulink environment)         | Reported             |

### III. MATHEMATICAL MODELING OF DEXTER-6 PRO

To start the modeling and analysis of Dexter-6 Pro, a detailed 3D model was created in SolidWorks. The CAD model made it possible to obtain important physical properties needed for

dynamic analysis, including the mass, center of mass, and moments of inertia for each link, calculated using SolidWorks' Mass Properties feature. The manipulator configuration has six revolute joints and a fixed base, making up a complete 6-DOF setup. Figure 2. shows the schematic of the Dexter-6 Pro robot in SolidWorks.

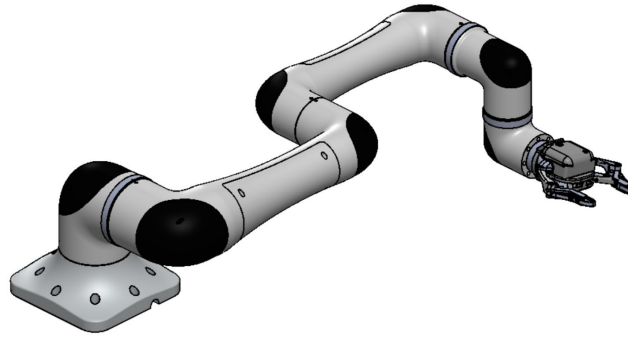


Fig.2 SolidWorks Model of the Dexter-6Pro Manipulator Showing the Fixed Base and Six Revolute Joints.

#### A. Kinematics

The kinematic model of the Dexter-6 Pro robotic arm was established with the Denavit–Hartenberg (D-H) convention. Forward and inverse kinematics were both analytically obtained. Inverse kinematics gave rise to closed-form

solutions for joint variables, which were subsequently verified by numerical simulations for accuracy. The DH parameters consisting of the link lengths, twist angles, offsets, and joint angles, of Dexter-6 Pro for the specified joint frames in Figure 3, are presented in Table II.

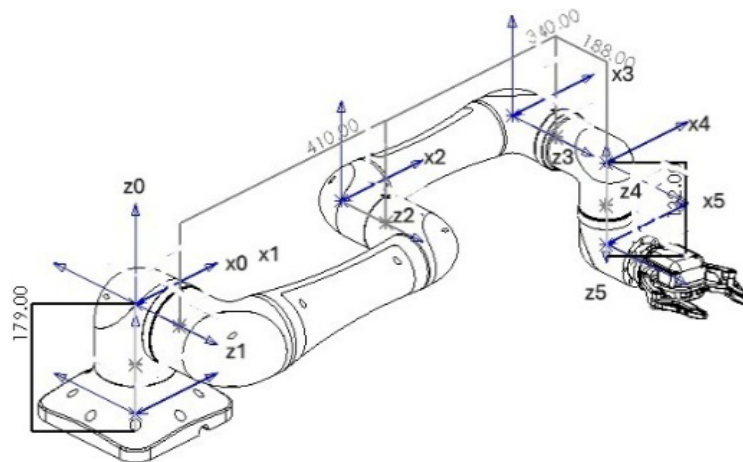


Fig.3 Coordinate frame assignment of 6-DOF Robotic Manipulator using the DH convention

TABLE II DH PARAMETERS OF DEXTER-6 PRO

| i | a   | $\alpha$ | d   | $\Theta$   |
|---|-----|----------|-----|------------|
| 1 | 0   | $\pi/2$  | 179 | $\theta_1$ |
| 2 | 410 | 0        | 0   | $\theta_2$ |
| 3 | 340 | 0        | 0   | $\theta_3$ |
| 4 | 0   | $\pi/2$  | 188 | $\theta_4$ |
| 5 | 0   | $-\pi/2$ | 132 | $\theta_5$ |
| 6 | 0   | 0        | 0   | $\theta_6$ |

### B. Forward Kinematics

A key element of robotic modeling is forward kinematics, which uses a set of joint variables to determine the orientation and position of a robot end-effector in the workspace. This computation involves creating homogeneous transformation matrices for every link using the Denavit-Hartenberg (DH) convention for a 6-DOF revolute-jointed manipulator, like the UR5 or the simulated arm shown in Figure 3. The transformation from the end-effector frame to the base frame  $T^0_6$  is achieved by multiplying a sequence of homogeneous transformation matrices [9] [8]:

$$T^0_6 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The components of the orientation matrix are represented by  $r_{ij}$ , while the end-effector coordinates,  $p_x, p_y, p_z$ , are shown by equations. The forward kinematics model [9] provides the mathematical basis for arm motion modeling in programs like Simscape Multibody and MATLAB/Simulink, as well as for inverse kinematics, trajectory planning, and real-time control integration. The transformation has been validated through both symbolic computation and 3D simulation.

### C. Inverse Kinematics

The method of calculating joint variables, or the angles of revolute joints, to achieve a particular end-effector position and orientation for the robotic manipulator is known as inverse kinematics. Utilizing the manipulator's geometric configuration and structural characteristics based on its Denavit-Hartenberg (DH) parameters, the closed-form analytical solution was obtained for a 6-DOF robotic arm. [11] This approach facilitates rapid computation and real-time operation, particularly when supplemented with MATLAB simulations for cross-pose verification of joint solutions.

$$A = p_y - d_6 * r_{23} \quad (2)$$

$$B = p_x - d_6 * r_{13} \quad (3)$$

$$C = c_1 * r_{11} + s_1 * r_{21} \quad (4)$$

$$D = c_1 * r_{22} - s_1 * \quad (5)$$

$$E = s_1 * r_{11} - c_1 * r_{21} \quad (6)$$

$$F = c_5 * c_6 \quad (7)$$

$$\theta_1 = \pm \text{atan2} \left( \sqrt{B^2 + (-A)^2 - d_4^2}, d_4 \right) + \text{atan2}(B, -A) \quad (8)$$

$$\theta_5 = \pm \text{atan2}(\sqrt{E^2 + D^2}, s_1 * r_{13} - c_1 * r_{23}) \quad (9)$$

$$\theta_6 = \text{atan2} \left( \frac{D}{s_5}, \frac{E}{s_5} \right) \quad (10)$$

$$\theta_{234} = \text{atan2}(r_{31} * F - s_6 * C, FC + s_6 * r_{31}) \quad (11)$$

$$Kc = c_1 * p_x + s_1 * p_y - s_{234} * d_5 + c_{234} * s_5 * d_6 \quad (12)$$

$$Ks = p_z - d_1 + c_{234} * d_5 + s_{234} * s_5 * d_6 \quad (13)$$

$$c_3 = \frac{K_s^2 + K_c^2 - a_2^2 - a_3^2}{2 * a_2 * a_3} \quad (14)$$

$$s_3 = \sqrt{1 - c_3^2} \quad (15)$$

$$\theta_3 = \pm \text{atan2}(s_3, c_3) \quad (16)$$

$$\theta_2 = \text{atan2}(K_s, K_c) - \text{atan2}(s_3 * a_3, c_3 * a_3 + a_2) \quad (17)$$

$$\theta_4 = \theta_{234} - \theta_2 - \theta_3 \quad (18)$$

### D. Best Angles Selection For Inverse Kinematics

Because of the 6-DOF manipulator's inherent redundancy, different joint configurations may provide the same end-effector pose when solving the inverse kinematics for Dexter-6Pro. An evaluation approach based on pose reconstruction error and joint feasibility was used to methodically choose the best configuration. Let the inverse kinematics function yield a set of  $n=8$  possible joint angles  $\{\theta^{(i)}\}_{i=1}^8$ , where each  $\theta^{(i)} = [\theta_1^{(i)}, \theta_2^{(i)}, \dots, \theta_6^{(i)}]^T \in R^6$  and  $T_{target} \in SE(3)$  denote the desired end-effector transformation matrix. For each configuration  $\theta^{(i)}$ , the following evaluation procedure is applied:

1. The joint angles are used to compute the reconstructed end-effector pose:

$$T^{(i)} = \text{Forward-Kinematics}(\theta^{(i)}),$$

2. The Euclidean norm of the difference between the desired and reconstructed poses (position-only) [9] is calculated as:

$$e^{(i)} = \| p_{\text{target}} - p^{(i)} \|_2, \quad (19)$$

Where  $p_{\text{target}} \in R^3$  and  $p^{(i)} \in R^3$  are the position vectors extracted from  $T_{\text{target}}$  and  $T^{(i)}$ , respectively.

3. The configuration that minimizes the pose error is chosen out of all the valid ones:

$$\theta^* = \text{argmin } e^{(i)}, \quad (20)$$

This method guarantees that the final joint angle solution  $\theta^*$  stays practical within the robot's mechanical limitations while also achieving the intended end-effector. This approach is particularly beneficial in redundant configurations. [9].

### E. Dynamics Modeling Of Dexter-6 Pro

The Euler-Lagrange approach was employed to formulate the dynamic model of the manipulator with the gravitational, centrifugal, Coriolis, and inertial forces at every link.

MATLAB symbolic computation was employed in order to calculate the dynamic equations, enabling simulation-based analysis of joint response and torque demand along several different trajectories.

Coordinate frames and DH parameters are utilized in order to compute the homogeneous transformation matrices of succeeding links. These matrices describe each translational and rotational coupling between two consecutive joints. The transformation matrix has the following generic form. [10]

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (21)$$

Each  $T_i^{i-1}$  matrix is computed using the matching parameters  $\theta$ ,  $d$ ,  $a$ , and  $\alpha$  from the DH table. [4].

The complete transformation  $T_0^6$  from the base coordinate frame to the end-effector of the Dexter-6Pro manipulator is accomplished by gradually multiplying the different homogeneous transformation matrices that arise from the Denavit-Hartenberg parameters which is expressed as follows:

$$T_0^6 = T_0^1 \cdot T_1^2 \cdot T_2^3 \cdot T_3^4 \cdot T_4^5 \cdot T_5^6 \quad (22)$$

The Jacobian matrix  $J$  for the Dexter-6Pro manipulator is derived from the orientation and position of each joint frame to the base frame. The velocity of the end-effector in task space and the joint velocities is related by:

$$v_e = J(q)\dot{q} \quad (23)$$

The Jacobian matrix  $J(q) \in R_{6 \times 6}$  of the manipulator encapsulates the relationship between joint velocities and end-effector velocities. It consists of two main parts:  $J_v$  which represents the linear velocity component,  $J_w$  which accounts for the angular velocity component. The Jacobian matrix for a revolute joint is computed using the following expressions: [4]

$$J = \begin{pmatrix} z_0 \times (O_6 - O_0) & z_1 \times (O_6 - O_1) & z_2 \times (O_6 - O_2) \\ z_3 \times (O_6 - O_3) & z_4 \times (O_6 - O_4) & z_5 \times (O_6 - O_5) \end{pmatrix} \quad (24)$$

$$z_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, z_i = \begin{pmatrix} T_i^0(1,3) \\ T_i^0(2,3) \\ T_i^0(3,3) \end{pmatrix}, i=1, 2, 3, 4, 5; \quad (25)$$

$$O_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, O_i = \begin{pmatrix} T_i^0(1,4) \\ T_i^0(2,4) \\ T_i^0(3,4) \end{pmatrix}, i=1, 2, 3, 4, 5, 6; \quad (26)$$

The vectors  $z_1 z_2 z_3 z_4 z_5 z_6$  represent the first three components of the third column of the transformation matrices respectively. These were obtained through sequential multiplication of the individual joint transformation matrices. Likewise, the position vectors  $O_1 O_2 O_3 O_4 O_5$  and  $O_6$  correspond to the first three components of the fourth column of the transformation matrices respectively, and are computed based on the assigned coordinate frames.

Using this formulation, the Jacobian matrix is constructed sequentially by evaluating both the angular and linear components for each of the six joints. For the first joint, the angular velocity Jacobian is  $J\omega_1 = z_0$ , while the linear velocity Jacobian is  $Jv_1 = [0 \ 0 \ 0]^T$ . For the second joint, the angular component becomes  $J\omega_2 = z_1$ , and the linear component is calculated as

$$Jv_2 = z_0 \times (O_1 - O_0) \quad (27)$$

For the third joint,

$$J\omega_3 = z_2, \text{ and } Jv_3 = z_0 \times (O_2 - O_0) \text{ and } z_1 \times (O_2 - O_1) \quad (28)$$

For the fourth joint, the angular part is  $J\omega_4 = z_3$ , and the linear component consists of three cross products:

$$J\omega_4 = z_0 \times (O_3 - O_0), z_1 \times (O_3 - O_1), \text{ and } z_2 \times (O_3 - O_2) \quad (29)$$

For the fifth joint,

$$J\omega_5 = z_4 \text{ and } Jv_5 = z_0 \times (O_4 - O_0) + z_1 \times (O_4 - O_1) + z_2 \times (O_4 - O_2) + z_3 \times (O_4 - O_3) \quad (30)$$

Finally, for the sixth joint, the angular part is

$$J\omega_6 = z_5, \text{ and}$$

$$Jv_6 = z_0 \times (O_5 - O_0), z_1 \times (O_5 - O_1), z_2 \times (O_5 - O_2), z_3 \times (O_5 - O_3), \text{ and } z_4 \times (O_5 - O_4) \quad (31)$$

Using the homogeneous transformation matrices of the related links, each vector  $z_i$  and  $O_i$  is recovered. The linear Jacobian matrix  $Jv$  is stacked vertically on top of the angular Jacobian matrix  $J\omega$  to form the full Jacobian matrix  $J$ . The Jacobian matrix can also be generated by differentiating the location of the center of mass of each link to the joint variables, which is an alternative to the geometric method. This approach is especially useful for dynamic modeling, where it is crucial to calculate link velocity precisely. The position of each center of mass in the base frame is computed as:

$$p_{ci} = O_{i-1} + R_i p_{ci}^T \quad (32)$$

Here,  $p_{ci}^T$  denotes the local position vector of the center of mass of the  $i^{th}$  link, expressed in its own coordinate frame. The matrix  $R_i$  represents the rotation matrix of the  $i^{th}$  link (relative to the previous link), derived using the Denavit–Hartenberg (DH) convention, and  $O_i$  is the translation vector of the  $i^{th}$  link frame. The linear velocity Jacobian for each link is then obtained by:

$$J_{vi} = \partial p_{ci} / \partial q_i, \quad \text{for } i=1, 2, 3, 4, 5, 6 \quad (33)$$

Although several prior studies have addressed Jacobian derivation in dynamic modeling, the accurate computation of the center of mass positions and corresponding Jacobians remains a significant challenge. The Kufieta *et al.* [10] [13] technique, for instance (referred to as Method A), uses translations and rotation matrices that are directly represented in the base frame to calculate COM locations.

$$p_{ci} = O_{i-1} + R_{i-1}^0 p_{ci}^T, \quad \text{for } i=1, 2, 3, 4, 5, 6 \quad (34)$$

Where  $R_i^0$  is the rotation matrix from frame  $i$  to the base frame, and  $O_i$  is the location of frame  $i$  relative to the prior frame. Despite being simple, this method showed structural differences from simulation findings using Simulink's Multibody Toolbox. Discrepancies between the simulation environment and the analytical formulation were exposed via Jacobian output mismatches.

This study presents a more robust formulation (Method B) to get over these restrictions. Instead of changing the center of mass locations directly to the base, it does so recursively between subsequent frames. Positions are calculated as follows:

$$p_{c1} = O_0 + R_1 p_{c1}^T; p_{c2} = O_1 + R_2 p_{c2}^T; p_{c3} = O_2 + R_3 p_{c3}^T \\ p_{c4} = O_3 + R_4 p_{c4}^T; p_{c5} = O_4 + R_5 p_{c5}^T; p_{c6} = O_5 + R_6 p_{c6}^T$$

In this case,  $p_{ci}^T$  is stated in the local frame and recursively translated through intermediate coordinate systems, while  $R_i$  is the rotation matrix from frame  $i$  to frame  $i-1$ . The structural accuracy and appropriateness of this frame-consistent approach for dynamic modeling have been confirmed by a thorough validation against the Jacobian produced by MATLAB Simscape Multibody. The related GitHub repository [17] contains the full Jacobian-based derivation. The kinetic energy of the system is used to calculate the inertia matrix, which measures the manipulator's resistance to acceleration. It is provided by [5] for 6-link robot manipulator.

$$M(q) = \sum_{i=1}^6 \left[ J_{vi}^T m_i J_{vi} + J_{wi}^T R_i^0 I_{ni} R_i^{0T} J_{wi} \right] \quad (35)$$

Let  $m_i$  denote the mass of the  $i^{th}$  link. The terms  $J_{vi}$  and  $J_{wi}$  represent the linear and angular Jacobians of the  $i^{th}$  link, each having dimensions  $3 \times 6$ . The rotation matrix that converts vectors from the local coordinate frame of link  $i$  to the base frame is indicated here by  $R_i^0$ . The symbol  $I_{ni}$  refers to the inertia tensor of the  $i^{th}$  link about its center of mass,

expressed in its local frame and having dimensions  $3 \times 3$ . The rotation matrix from link 6's frame to the base, or  $R_6^0$ , is created by multiplying the rotation matrices of each joint up to joint 6:

$$R_6^0 = R_1^0 R_2^1 R_3^2 R_4^3 R_5^4 R_6^5 \quad (36)$$

This transformation ensures that the inertia tensors are correctly represented in a common reference frame. The matrix  $C(q, \dot{q}) \in \mathbb{R}^{n \times n}$  signifies the total effect of Coriolis and centrifugal forces, arising from velocity-dependent coupling among joints of the robot manipulator. To obtain the Coriolis matrix from the inertia matrix  $M(q)$ , we use the Christoffel symbols of the first kind, which are defined as [5]:

$$C_{ijk} = \sum_{i=1}^n \frac{1}{2} \left( \left( \partial \left( \frac{M_{ij}}{q_k} \right) + \partial \left( \frac{M_{jk}}{q_i} \right) - \partial \left( \frac{M_{ik}}{q_j} \right) \right) q_i \right) \quad (37)$$

In this case,  $M_{ij}$  is the  $i^{th}$  and  $j^{th}$  element of the inertia matrix, and  $C_{ijk}$  is an intermediate coefficient related to the  $i^{th}$  row and  $j^{th}$  column of the inertia matrix about the joint variable  $q_k$ .

Particularly in manipulators with vertically oriented joints, the gravitational force vector  $G(q) \in \mathbb{R}^n$  plays a considerable role in the joint torques by accounting for the influence of gravity acting on the center of mass of each link. The following expression is used to calculate each link's center of mass position, stated in the base coordinate frame:

$$p_{ci}^{base} = O_i + R_i^0 p_{ci} \quad (38)$$

Which is the position vector of the center of mass of link  $i$  in the base frame, where  $p_{ci}$  is the local location of the center of mass in the frame of link  $i$ , and  $O_i$  is the origin of the frame connected to link  $i-1$ . This formula ensures that for every manipulator arrangement, the vertical height of each center of mass, which is necessary to determine the gravitational potential energy, is computed exactly. The potential energy is subsequently calculated as:

$$P(q) = m_i g p_{ci}^{base} \quad (39)$$

The following formula is used to get the components of the gravity vector  $g$  from the system's potential energy  $P(q)$ :

$$G(q) = \partial P(q) / \partial q \quad (40)$$

The overall joint torques  $\tau \in \mathbb{R}^n$  to propel the manipulator can be computed using the classical Lagrange-Euler dynamic equation once the dynamic parts of the individual components, namely the inertia matrix  $M(q)$ , the Coriolis/centrifugal matrix  $C(q, \dot{q})$ , and the gravity vector  $G(q)$ , have been symbolically obtained:

$$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) = \tau \quad (41)$$

A methodical and organized framework for modeling the dynamics of robotic manipulators is provided by the Lagrange-Euler formulation. The method enables accurate joint torque prediction through derivation of the inertia, Coriolis, and gravity factors.

#### F. Review Of Control Strategies For Dynamic Validation

Model-based control strategies provide robust performance when system dynamics are known. In the current research, a PD controller was chosen due to its prior proven effectiveness and consistency in robotic manipulators [5]. This design successfully linearizes the closed-loop dynamics into a decoupled set of second-order systems, allowing precise trajectory following and stable error dynamics [6] [7]. The control law used is:

$$M(q)\ddot{q}_d + C(q, \dot{q})\dot{q}_d + G(q) + K_D(\dot{e}) + K_P(e) = \tau \quad (42)$$

where  $\tau$  = torque,  $q_d$ ,  $\dot{q}_d$ ,  $\ddot{q}_d$  denote the desired joint positions, velocities, and accelerations, respectively. Position error:  $e = q_d - q$ , Velocity error:  $\dot{e} = \dot{q}_d - \dot{q}$ , here  $q$  and  $\dot{q}$  represent the actual joint positions and velocities. The gain matrices,  $K_P$ ,  $K_D$  and dynamic model terms  $M(q)$ ,  $C(q, \dot{q})$ , and  $G(q)$  are derived symbolically via the Euler-Lagrange formulation. Under ideal modeling conditions, the resulting closed-loop error dynamics take the form: [12]

$$\ddot{e} + K_D(\dot{e}) + K_P(e) = 0 \quad (43)$$

ensuring exponential convergence to the desired trajectory.

## IV. SIMULATION IN MATLAB

### A. Validation Methodology

The procedure was methodically carried out in MATLAB/Simulink to verify the accuracy and applicability of the dynamic model of the Dexter-6 Pro manipulator. MATLAB Simscape Multibody was used as a reference for physics-based simulation. This allowed direct comparison of symbolic dynamics and Simscape generated responses, ensuring correctness before hardware implementation [14]

### B. Inverse Kinematics Implementation

Inverse kinematics was implemented with a MATLAB Function block in Simulink [15]. Due to multiple possible joint configurations ( $\theta_1$ ,  $\theta_3$ ,  $\theta_5$  sign variations), eight possible solutions were generated and evaluated. The configuration minimizing pose error was selected and visualized using the Simscape Mechanics Explorer to ensure accurate workspace execution (Figure 4 and Figure 5).

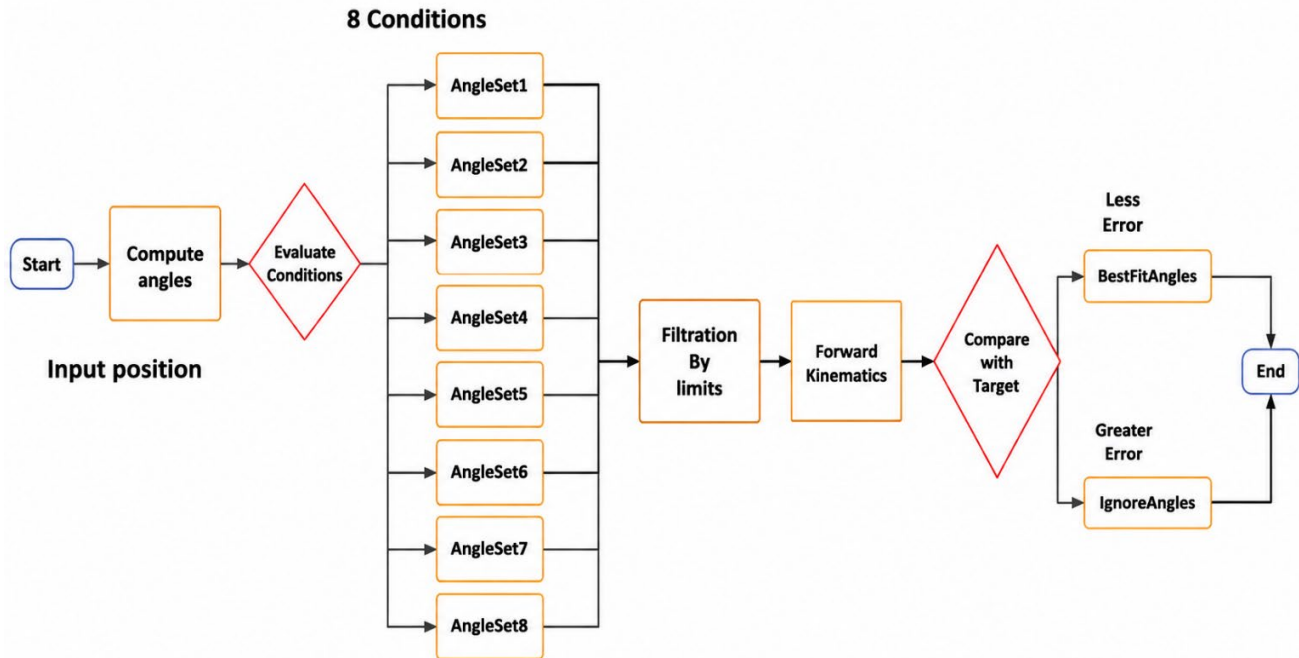


Fig.4 The step-by-step Process for Implementing Inverse Kinematics

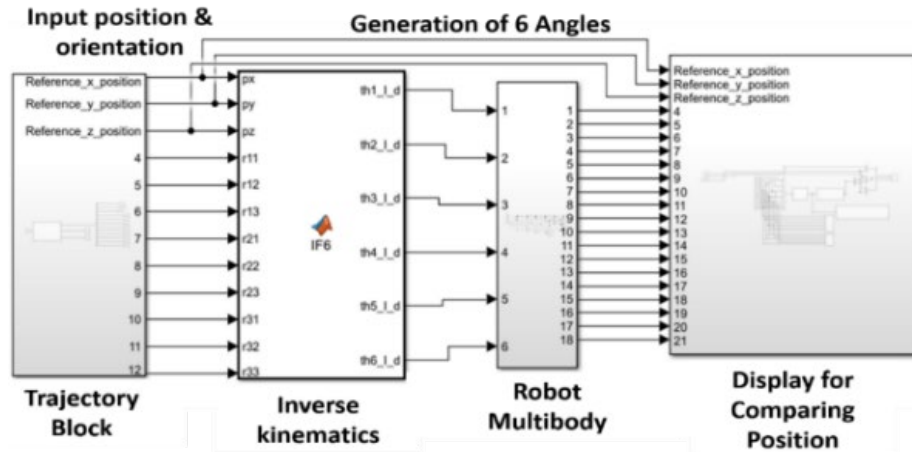


Fig.5 Inverse kinematics Simulink Model of the Robotic Manipulator, used to compute joint angles for a given end-effector path.

### C. Inverse Dynamics And Control Simulation

Using a hybrid approach that combined numerical simulation and symbolic torque calculation, dynamic simulation was performed. A major challenge in this procedure is assessing dynamic model terms, especially the inertia matrix  $M(q)$ , the gravity vector  $G(q)$ , and the Coriolis matrix  $C(q, \dot{q})$ , which can become computationally expensive because of their symbolic character. The Symbolic Math Toolbox [16] was used to symbolically deduce  $M(q)$ ,  $C(q, \dot{q})$ , and  $G(q)$  from the Euler–Lagrange formulation in order to prevent this. To enable it in real-time, a number of computational methods were used:

1. Symbolic simplification was used to eliminate redundant terms before code generation.
2. Conversion to numeric functions via `matlabFunction` enabled substitution of joint states ( $q, \dot{q}$ ) at runtime in Simulink.

Benchmarking showed that a direct symbolic evaluation in Simulink required approximately 120 ms per time step, whereas the compiled numeric functions reduced computation to under 5 ms, making the framework suitable for real-time simulation.

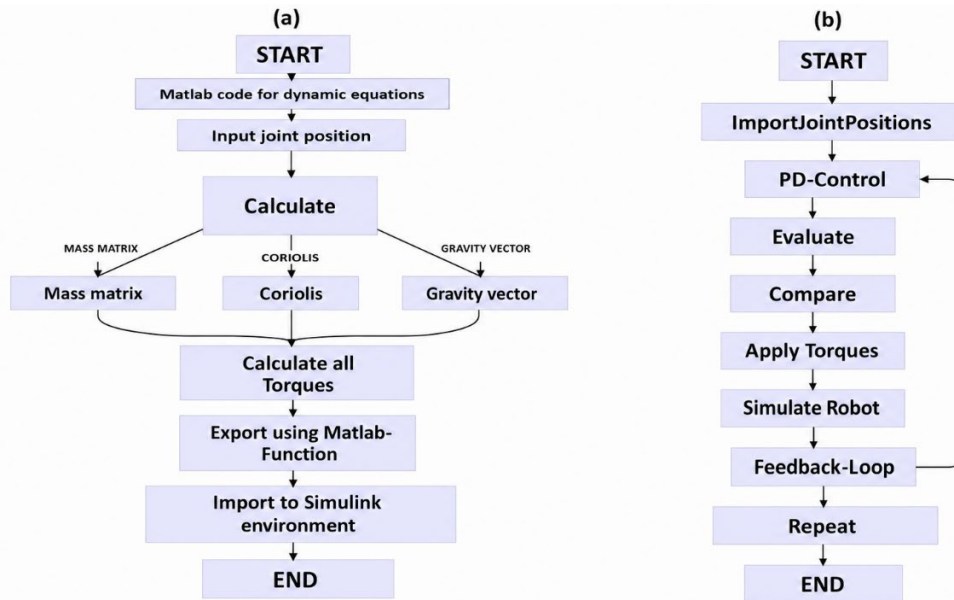


Fig.6 Flowchart illustrating the step-by-step process for implementing inverse dynamics (a) Workflow for Generating Joint Torque Function in MATLAB for Simulink Integration, (b) PD-Based Joint Torque Control Flow for Robot Simulation

In subfigure (a) of Figure 6, torque equations are derived symbolically using the Euler–Lagrange formulation and converted into executable functions via a MATLAB Function

block. Subfigure (b) shows the PD-based control loop, where the computed torques are applied to the Simscape Multibody model for real-time closed-loop simulation.

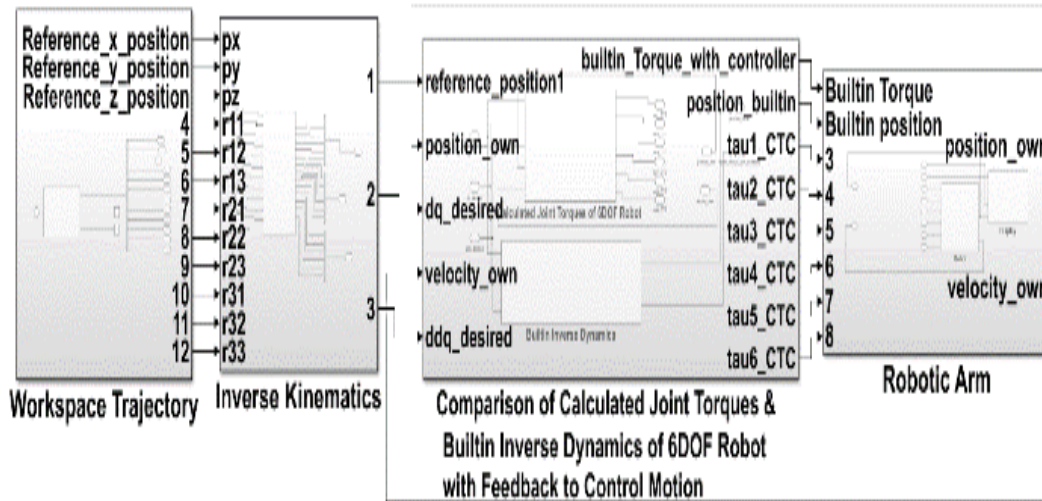


Fig.7 Inverse dynamics Simulink model of the 6-DOF robotic manipulator, implementing motion control using workspace trajectory tracking for a welding path, with feedback regulation for accurate joint torque computation.

It Shows the Simulink-based validation structure used to evaluate the dynamic model of the Dexter-6 Pro manipulator with a Torque Control scheme. The physical prototype presented in Figure 1 was employed to verify the link geometry, assembly constraints, and motion feasibility of the simulation model. Although this research concentrates mainly on simulation and modeling, the availability of the fabricated prototype guarantees the physical realizability of the model proposed.

## V. RESULTS

### A. Numerical Validation Of Jacobian Matrices

To verify the correctness of kinematic modeling, we compared the obtained Jacobian matrices from two methodologies for some of the manipulator's links. Figure 8 shows the Jacobians of the third and fourth links calculated via the built-in Get Jacobian Simulink block. On the other hand, Figure 9 shows the respective Jacobians calculated using hand-coded symbolic manipulations in MATLAB, created with the manipulator's DH parameters and checked transformation matrices.

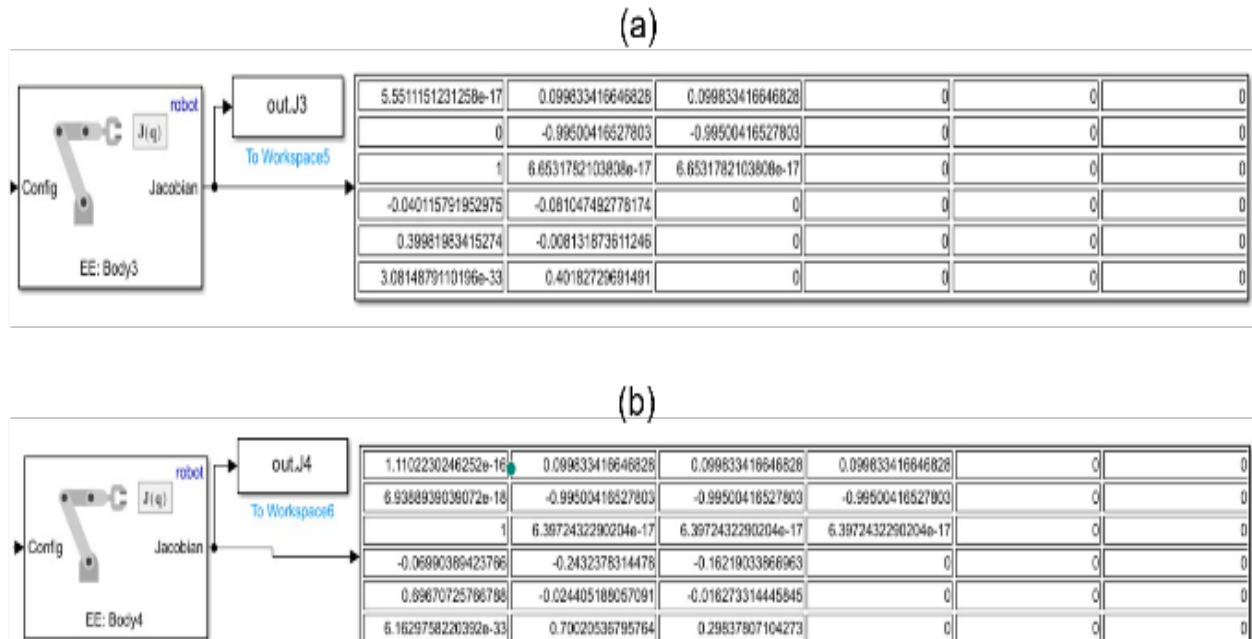


Fig.8 Comparison of the Jacobian matrices for the third (a) and fourth (b) bodies of the manipulator, with respect to the base frame, using built-in Get Jacobian block in Simulink.

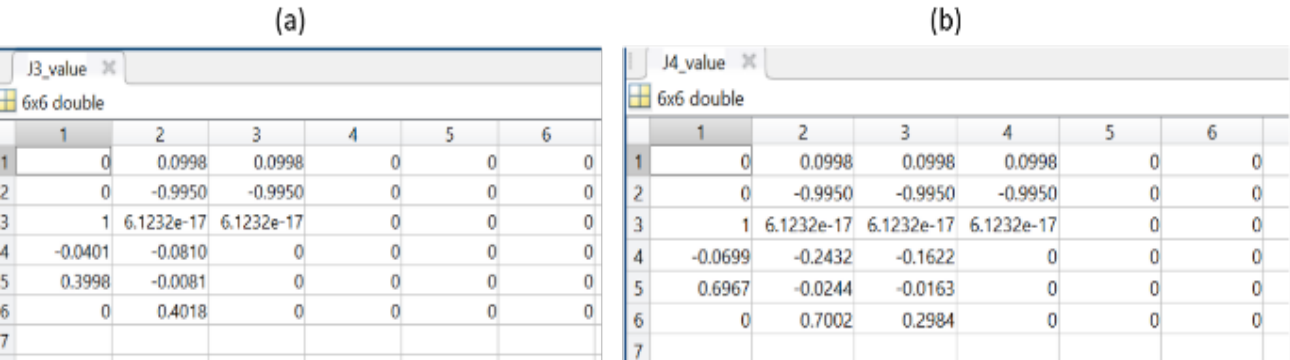


Fig.9 Comparison of the Jacobian matrices for the third (a) and fourth (b) bodies of the manipulator, with respect to the base frame, using hand-derived symbolic computation in MATLAB.

B. Graphical Analysis Of Kinematic And Dynamic Models

Figure 10 illustrates the 6-DOF inverse kinematics solution's performance by comparing the reference (target) end-effector

positions and the measured (actual) positions along the x, y, and z axes as a function of time. The measured path tracks the reference pick-and-place path closely.

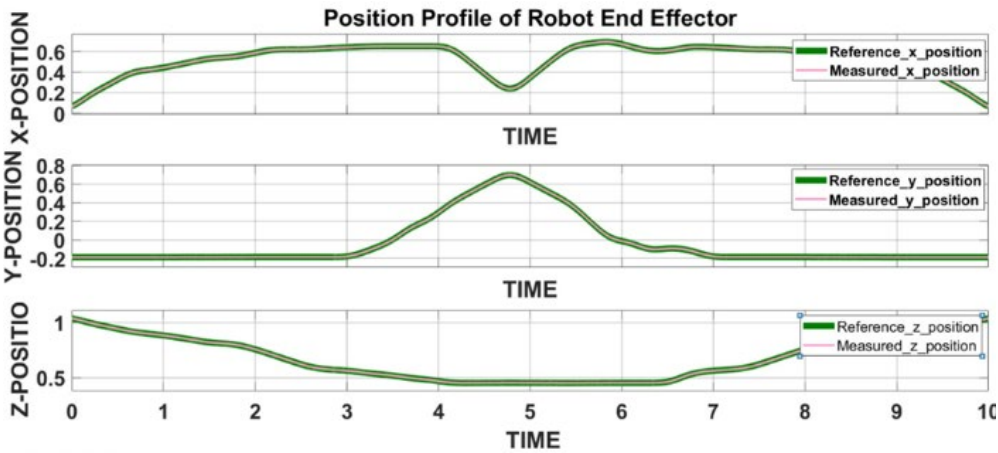


Fig.10 Visualization from the Simscape Simulation Comparing Target and Actual end-effector Trajectories Over Time

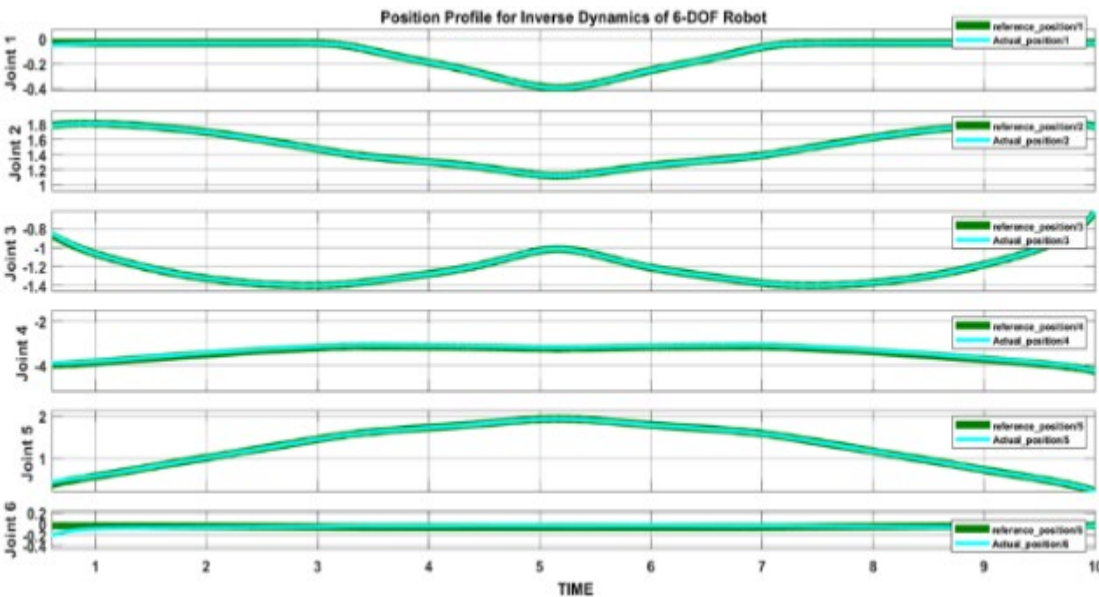


Fig.11 Graphical comparison between the input reference position and the achieved end-effector position, demonstrating the precise tracking capabilities of the dynamic modeling framework for the Sim-mechanic model of Dexter-6 Pro.

Figure 11 compares the reference end-effector trajectory with the simulated trajectory obtained from inverse dynamics. The close alignment indicates the dynamic model's correctness and compatibility. Minor deviations arise from numerical approximations and model simplifications. Figure 12 shows a comparison between the joint torques that can be computed

using the manually developed inverse dynamics equations and those generated using the built-in inverse dynamics block in the Robotic System Toolbox. The torque profiles show a general agreement across all six joints, suggesting that the manually developed dynamic model works reasonably well in comparison with the toolbox-based implementation.

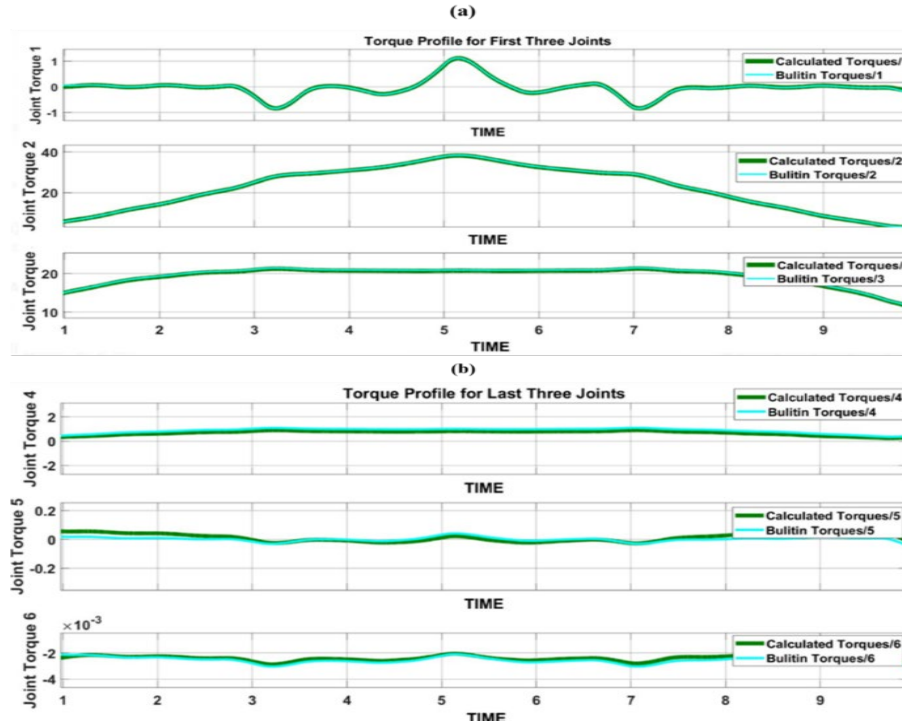


Fig.12 Graphical comparison of joint torques for the Simmechanic model using hand-derived inverse dynamics equations and those generated by the Inverse Dynamics block from MATLAB's Robotics System Toolbox. (a) Torques for joint 1 to joint 3. (b) Torques for joint 4 to joint 6.

## VI. CONCLUSION AND FUTURE WORK

This paper presented the development of Dexter-6 Pro, a 6-DOF lightweight robot manipulator designed for industrial and laboratory automation. A complete symbolic dynamic model was derived using the Euler-Lagrange formulation, and inverse dynamics and kinematics were implemented in MATLAB. To fully validate the created model, a Simulink-based platform was used to implement a Torque Control system demonstrating stable trajectory monitoring and convergence of the end-effector errors. The results of modeling and simulation confirm the analytical validity of the intended framework. Compared with existing research (Table I), the combination of symbolic formulation and control validation presents a more comprehensive and practically applicable solution. Hardware verification is outside the scope at present, but the results given prove the system to be ready for real-time control experiments. Dexter-6Pro shows strong potential as a low-cost, locally fabricated robotic platform for advancing automation in Pakistan. Future work will focus on hardware-based validation and performance assessment under real-world uncertainties such as friction, sensor noise, and payload variations. Additionally, adaptive and robust control strategies will be explored to enhance reliability in diverse operating environments.

## CODE AVAILABILITY

The overall kinematic and dynamic modeling code developed for Dexter-6Pro is available at:  
<https://github.com/MechatronicsEngineer21/Dexter6Pro-Robotic-Arm-Kinematics-Dynamics.git>

### Declaration of Conflicting Interests

The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

### Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

### Use of Artificial Intelligence (AI)-Assisted Technology for Manuscript Preparation

The author(s) utilised generative AI technologies solely to improve language clarity and refine grammar. No images were generated, manipulated, or altered using AI-assisted technologies.

## REFERENCES

- [1] Mohamed S. Elhadidy, Waleed S. Abdalla, Alaa A. Abdelrahman, S. Elnaggar, Mostafa Elhossaini, "Assessing the accuracy and efficiency of kinematic analysis tools for six-DOF industrial manipulators: The KUKA robot case study," AIMA Press, vol. 9, no. 6, 2024.
- [2] Jianfei Xu, Zhou Xu, Lei Yang, Zhengyong Tao, Min Qu, Hui Wu, Mingming Wu, and Dongdong Ye, "Joint kinematics and dynamics

- analysis of a 6-DOF loading and unloading industrial robot based on ADAMS and ANSYS," *Mechanical Sciences*, vol. 16, no. 2, 2025.
- [3] M. R. Abbas, M. Ahsan and J. Iqbal, "Experimental development of lightweight manipulators with improved design cycle time that leverages off-the-shelf robotic arm components," *PLoS ONE*, vol. 19, no. 7, p. e0305379, 2024.
- [4] M. H. Barhaghtalab, V. Meigoli, M. R. Golbahar Haghighi, S. A. Nayeri and A. Ebrahimi, "Dynamic analysis, simulation, and control of a 6-DOF IRB-120 robot manipulator using sliding mode control and boundary layer method," *Journal of Central South University*, vol. 25, no. 12, p. 2885–2897, 2018.
- [5] P. M. Kebria, S. Al-wais, H. Abdi and S. Nahavandi, "Kinematic and dynamic modelling of UR5 manipulator," in *IEEE International Conference on Systems, Man, and Cybernetics*, Budapest, 2016.
- [6] B. Siciliano, L. Sciacivco, L. Villani and G. Oriolo, *Robotics: Modelling, Planning and Control*, Springer, 2010.
- [7] M. W. Spong, S. Hutchinson and M. Vidyasagar, "Robot Modeling and Control," 2006.
- [8] J. Villalobos, I. Y. Sanchez and F. Martell, "Statistical comparison of Denavit-Hartenberg based inverse kinematic solutions of the UR5 robotic manipulator," in *International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME)*, Mauritius, 2021.
- [9] J. Villalobos, I. Y. Sanchez and F. Martell, "Singularity Analysis and Complete Methods to Compute the Inverse Kinematics for a 6-DOF UR/TM-Type Robot," *Robotics*, vol. 11, no. 6, p. 137, 2022.
- [10] K. A. Kufieta, *Force estimation in robotic manipulators: Modeling, simulation and experiments*, Norwegian University of Science and Technology, 2014.
- [11] H. A. Moreno, R. A. Ramirez-Mendoza and I. A. Banfield, "Advances in Automation and Robotics Research," vol. Volume 123, Cham, Springer Nature Switzerland AG, 2022.
- [12] IEEE, "2016 IEEE International Conference on Systems, Man, and Cybernetics: Conference Proceedings," in *IEEE*, 2016.
- [13] kkufieta. [https://github.com/kkufieta/ur5\\_modeling\\_force\\_estimate](https://github.com/kkufieta/ur5_modeling_force_estimate).
- [14] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd, Ed., Pearson, 2005.
- [15] M. Inc., "MATLAB Function block," 2023. <https://www.mathworks.com/help/simulink/slref/matlabfunction.html>.
- [16] M. Documentation, "Get Started with Symbolic Math Toolbox," [https://www.mathworks.com/help/symbolic/getting-started-with-symbolic-math-toolbox.html?s\\_tid=CRUX\\_lftnav](https://www.mathworks.com/help/symbolic/getting-started-with-symbolic-math-toolbox.html?s_tid=CRUX_lftnav).
- [17] MechatronicsEngineer21, "Dexter6Pro Robotic Arm: Kinematics and Dynamics," <https://github.com/MechatronicsEngineer21/Dexter6Pro-Robotic-Arm-Kinematics-Dynamics>.