mathematics

MDPI

Article

Quantum Particle Swarm Optimization (QPSO)-Based Enhanced Dynamic Model Parameters Identification for an Industrial Robotic Arm

Mehdi Fazilat and Nadjet Zioui *

Department of Mechanical Engineering, University of Quebec at Trois-Rivières, Trois-Rivières, QC G8Z 4M3, Canada; mehdi.fazilat@uqtr.ca
* Correspondence: nadjet.zioui@uqtr.ca

Abstract

Accurate parameter identification in dynamic models of robotic arms is essential for performing high-performance control and energy-efficient procedures. However, classic methods often encounter difficulties when modeling nonlinear, high-dimensional systems, particularly in the presence of real-world uncertainties. To address these challenges, this study focuses on identifying mass center positions and inertia matrix elements in a six-jointed industrial robotic arm and comparing the influence of optimized algorithms: the classical Particle Swarm Optimization (PSO) and the Quantum-behaved Particle Swarm Optimization (QPSO). The robot's kinematic model was validated by comparing it with actual motion data, utilizing a high-precision neural network to ensure accuracy before conducting a dynamic analysis. A comprehensive dynamic model was created using Computer-Aided Optimization (CAO) in SolidWorks Premium 2023 to simulate realistic mass parameters, thereby validating the model's reliability in a practical setting. The real (Referenced) and optimized dynamic models of the robot arm were validated using trajectory tracking simulations under sliding mode control (SMC) to assess the impact of the optimized model on the robot's performance metrics. Results indicate that QPSO estimates inertia and mass center parameters with Mean Absolute Percentage Errors (MAPE) of 0.76% and 0.43%, outperforming PSO significantly and delivering smoother torque profiles and greater resilience to external disturbances.

Keywords: industrial robots; nonlinear dynamics; nonlinear control systems; parameter estimation; swarm optimization; quantum-behaved particle swarm optimization

MSC: 34A34; 68T07; 03G12

check for updates

Academic Editor: Cristian Lazureanu

Received: 30 July 2025
Revised: 11 August 2025
Accepted: 13 August 2025
Published: 16 August 2025

Citation: Fazilat, M.; Zioui, N. Quantum Particle Swarm Optimization (QPSO)-Based Enhanced Dynamic Model Parameters Identification for an Industrial Robotic Arm. *Mathematics* **2025**, *13*, 2631. https://doi.org/10.3390/math13162631

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Context of the Study

In the rapidly growing specialization of industrial robotics, precise dynamic modeling is essential for achieving high-precision motion control, ensuring reliable performance, and enhancing energy efficiency. Accurately identifying dynamic parameters becomes increasingly important as robotic systems become more complicated and autonomous, particularly in articulated arm applications. Among these parameters, the elements of the inertia matrix and the positions of mass centers are especially influential. These factors characterize mass distribution and rotational inertia, directly affecting the robot's response

to applied forces under diverse operational conditions. Accurate estimation of these parameters guarantees smooth and stable motion and supports the development of advanced control algorithms, fault diagnosis systems that are critical for system maintenance, and predictive maintenance strategies. Effectively modeling these characteristics is essential for improving robotic systems' intelligence, adaptability, and robustness in real-world industrial environments [1–3].

Traditional parameter estimation methods in the dynamics of robotic arms have historically been founded on analytical approaches such as Least Squares (LS), Recursive Least Squares (RLS), and the Newton-Raphson method. While these techniques are computationally efficient and well-suited to linear systems or scenarios with clearly defined mathematical models, they face considerable limitations in the presence of noise and uncertainty. However, the field has not remained stagnant. The emergence of alternative strategies, such as probabilistic estimation techniques like the Kalman Filter (KF), Extended Kalman Filter (EKF), and Particle Filter, and Commonness Domain Methods and Grey Box Modeling, has enhanced estimation accuracy under dynamic measurement requirements. These strategies represent progress and evolution in the field, facilitating flexible identification based on specific operational assumptions. Despite their extensive application, these classical techniques show notable deficiencies when addressing the complexity of robotic systems. The dynamic modeling of industrial robotic arms often involves highly nonlinear behaviors, varying loading conditions, and high-dimensional parameter space situations in which these methods work. For example, the linearity assumptions inherent in LS or RLS can result in systematic errors. At the same time, Kalman-based filters rely on accurately characterized noise statistics and may experience divergence when applied to poorly modeled systems. Additionally, although the Newton-Raphson technique can be effective for specific iterative solutions, it frequently encounters convergence issues in the presence of non-convexities or abrupt parameter changes. These challenges underscore the need for advanced strategies that are robust and precise for parameter identification in modern robotic systems operating under real-world variability and dynamic uncertainty [4–7].

1.2. Related Works

Given these challenges, this research identifies the inertia matrix and mass center parameters of articulated robotic arms operating under nonlinear, high-dimensional, and uncertain conditions. Although traditional estimation techniques may perform well under idealized conditions, they frequently prove inadequate in intricate industrial settings represented by dynamic variability and operational uncertainty. In recent advancements, the accurate identification of inertia matrix and mass center parameters has been particularly enhanced through Particle Swarm Optimization and other refined methodologies, with direct implications for practical applications. For instance, Xu et al. [8] explored PSOs application to space robotics, focusing on dynamically identifying inertia parameters through equivalent dynamics. Similarly, Bingül and Karahan [9] utilized a combination of PSO and Least Squares (LS) methods to perform dynamic identification on the Staubli RX-60 robot, thus showcasing a hybrid approach that leverages PSOs optimization capabilities with the precision of LS. Zhong et al. [10] also enhanced dynamic parameter identification by integrating an enhanced PSO algorithm with a comprehensive excitation trajectory, specifically for the 6R robotic arm, enabling more accurate and reliable parameter estimates. Additionally, Leboutet et al. [11] provided a comprehensive survey of inertial parameter identification techniques, positioning PSO as a critical tool in overcoming the challenges associated with nonlinear dynamics and ensuring robust parameter identification. These studies collectively highlight the development and application of advanced optimization

techniques like PSO in enriching the fidelity and efficiency of robotic system identification, with direct implications for the evolution of advanced robotic systems.

QPSO, which enhances the conventional PSO approach, is an advanced methodology utilized in research to handle complicated, high-dimensional parameter spaces like robotic arm modeling. It can effectively avoid local optima and seek global solutions, facilitating real-time data processing through a distributed computational approach, which is essential for overcoming the limitations of current parameter identification methods [12]. QPSO particularly improves solution quality, convergence speed, and adaptability across various engineering applications, including improving precision [13], optimizing both kinematic and non-kinematic parameters of five-bar parallel robots [14], solving inverse kinematics problems [15,16], and refining trajectory tracking control, consistently outperforming conventional methods for the robotic arms [17].

Parameter identification of Robot Manipulators has been the focus of a large number of studies exploiting PSO and its hybrid variants in the literature. These studies can be considered a breakthrough in the field, and all of them have achieved substantial enhancements in both convergence rate and accuracy, as well as robustness. For example, a classical PSO algorithm was used to identify the complete inertial parameters of a 6-DOF space manipulator attached to the satellite base, with a simple dynamic model and moment equations of motion in microgravity [8]. A hybrid developmental method that utilized PSO in conjunction with Differential Evolution DE for the parameter estimation of a simulated 3-DOF cylindrical robot, benefiting from DEs ability to search the domain space as well as PSOs speed of convergence [9]. Furthermore, research conducted has provided both theoretical insights into PSO and practical applications. For instance, a variant of PSO, similar to one that introduced dynamic time-varying acceleration coefficients to improve parameter estimation of a 3-DOF planar manipulator, achieves better control over convergence in the iterations [18]. Similarly, in another article, a developed PSO based on time-varying inertia weights and learning factors is used to estimate the parameters of the simulated 3-DOF robotic arm cited in [19]. These applications of PSO in practice have proven its efficiency. Additionally, research on the original PSO and its modifications has consistently confirmed its stability. For example, an Adaptive PSO algorithm adjusting the parameters of the algorithm with the swarm dynamics evolved was utilized to solve for the parameters of a 3-DOF manipulator and showed better accuracy and stability [20]. A hybrid algorithm based on PSO and GWO, which emulates the leadership hierarchy and hunting behavior of grey wolves, was developed to identify the dynamic parameters of a 3-DOF robot arm in SimMechanics and outperformed the two separate algorithms [21]. These consistent findings demonstrate the strength of PSO in a wide range of applications. The application of Quantum-behaved Particle Swarm Optimization to a similar 3-DOF manipulator model, which enhances the global searching ability by implementing principles derived from the theory of quantum mechanics, yields better robustness and identification accuracy compared to classical PSO [22]. Finally, the comparison study of PSO with some powerful optimization algorithms in the case of a 3-DOF manipulator showed that PSO yields highly competitive solution quality and noise tolerance results compared to other approaches [23].

1.3. Research Gap and Motivation

According to extensive reviews of the existing literature, notable gaps persist in advanced methods for parameter estimation within the dynamic models of various optimization algorithms. While techniques such as PSO have indicated promise, they frequently encounter difficulties in high-dimensional spaces and under rapidly changing conditions [24–27]. These limitations underscore the need for strategies that effectively integrate advanced computing technologies with optimization algorithms to enhance the

robustness of dynamic modeling. Moreover, there is a need for integration within comprehensive modeling frameworks for industrial robotic arms. Traditional Particle Swarm Optimization and its variants have been employed primarily to identify parameters for low-degree-of-freedom or modular manipulators, often relying on simplified or simulated models. However, these approaches tend to focus on specific components, such as joint stiffness or rehabilitation arms, rather than offering a holistic solution for dynamic modeling. Although prior works have utilized QPSO to tune models of limited complexity, none have specifically addressed parameter estimation for a high-degree-of-freedom industrial manipulator by employing CAD-derived mass properties in conjunction with QPSO-based optimization. Importantly, no study has integrated the identified parameters into a robust and applicable framework for practical assignment to evaluate performance metrics against external disturbances. This gap regarding both the methodological coupling of QPSO with CAD-informed modeling and its validation in closed-loop control defines the innovative trajectory of the current work.

1.4. Key Contributions and Novelty of the Study

The primary objective of this research is to develop and validate a high-fidelity dynamic model for a six-degree-of-freedom industrial robotic arm by accurately determining its inertial and mass center parameters based on physical parameters derived from Computer-Aided Design models and to structure a reference model for the investigation. To achieve this, the study systematically employs and compares the performance of classical Particle Swarm Optimization with Quantum-behaved PSO algorithms for identifying these dynamic parameters under nonlinear, high-dimensional, and uncertain operating conditions. In addition to parameter identification, the optimized and referenced models are integrated into a motion planning framework based on Sliding Mode Control, allowing for a quantitative assessment of their impact on torque smoothness, energy consumption, and trajectory tracking accuracy in both nominal and disturbed scenarios.

The present research completes three key contributions. First, we extract and utilize physically accurate inertia and mass center parameters derived from CAD models of a real industrial robotic arm, moving away from the conventional reliance on simplified assumptions. Second, we develop a unified multi-stage framework that incorporates high-precision kinematic validation using a multilayer perceptron, quantum-behaved swarm-based dynamic parameter identification, and Sliding Mode Control for performance evaluation in practical trajectory tracking tasks, including scenarios involving disturbances. Lastly, we conduct a comprehensive comparative analysis between classical PSO and QPSO, illustrating the enhanced accuracy, energy efficiency, and torque smoothness achieved with QPSO, thereby establishing its practicality for real-world robotic control applications.

The novelty of this research originates from its thorough examination of previous studies that either depended on oversimplified dynamic assumptions or assessed heuristic optimization methods without rigid validation in an industrial context. First, this study meticulously extracts the mass and inertia parameters of a real six-degree-of-freedom industrial robot using SolidWorks-based CAD modeling (SolidWorks Premium 2023), thereby replacing the commonly used approximations found in similar works. Second, it presents a systematic comparison between classical Particle Swarm Optimization and Quantum-behaved PSO, showcasing the exceptional performance of QPSO in terms of convergence accuracy, torque stability, and energy efficiency. Third, unlike prior research that focused exclusively on either identification or control, this study integrates QPSO within a robust Sliding Mode Control framework, promoting a closed-loop evaluation under both nominal and disturbed trajectory tracking conditions. Finally, an experimented

high-fidelity multilayer perceptron (MLP) network is employed to validate the kinematic model, ensuring accuracy before proceeding with the dynamic analysis.

The rest of this article is structured as follows: Section 2 summarizes the modeling framework and optimization setup, encompassing kinematics, dynamics, and the CAD model, and elaborates on the formulation and tuning strategies of the PSO algorithms. Section 3 presents comparative results and analysis. Finally, Section 4 concludes the study and suggests future directions for optimization-integrated robotic control systems.

2. Materials and Methods

2.1. Robotic Arm Presentation

The study focuses on the ABB IRB 140 (M2004) robot, illustrated in Figure 1. This robot, featuring a six-axis articulated design, is extensively used across various industrial applications with flexible mounting options to adapt to various working conditions by supports an end-effector, including a payload weighing up to 5 kg with considerable working space, and the manipulator utilized in its unmodified, factory-default configuration to ensure results reflect standard performance and provide a reliable baseline for evaluation [7].



Figure 1. The ABB IRB 140 robot in mechatronic laboratory.

2.2. Methodological Framework

The modeling procedure for the ABB IRB-140 robotic arm is initiated with the development of the manipulator's kinematic models to establish the position and orientation of the end-effector. The provided kinematic models are rigorously validated against real-world motion data using a high-precision singularity-avoiding MLP, ensuring their accuracy and practical applicability before dynamic analysis [28,29]. The dynamic model of the robot was formulated using the Euler–Lagrange method to calculate joint torques and energy consumption and was enhanced by integrating mass center and inertia parameters extracted from SolidWorks (Premium Version 2023) CAD models (SolidWorks Premium 2023). These CAD-derived parameters were defined as the reference baseline for all subsequent analyses. The Computer-Aided Optimization (CAO) phase enabled high-fidelity design modeling to ensure accurate multi-body representation and energy profiling.

In the next stage, the optimization of these dynamic parameters was performed using two distinct algorithms: classical PSO and Quantum-behaved PSO. Both algorithms were applied to identical input spaces, namely, the CAD-derived reference parameters for inertia and mass centers. Their performance was assessed purely based on the internal mechanics of each method, with no difference in their input data or objective functions.

This approach ensures a fair comparison of their effectiveness in optimizing physically meaningful parameters.

The final phase of the research involved a comparative evaluation between the reference model and the optimized models obtained using PSO and QPSO. These were re-integrated into the dynamic simulation framework under a sliding mode control (SMC) strategy to analyze key performance indicators such as torque smoothness, energy consumption, and tracking accuracy. Figure 2 illustrates the complete data channel of the methodology, from initial CAD-based parameter extraction to performance validation under integrated control evaluation.

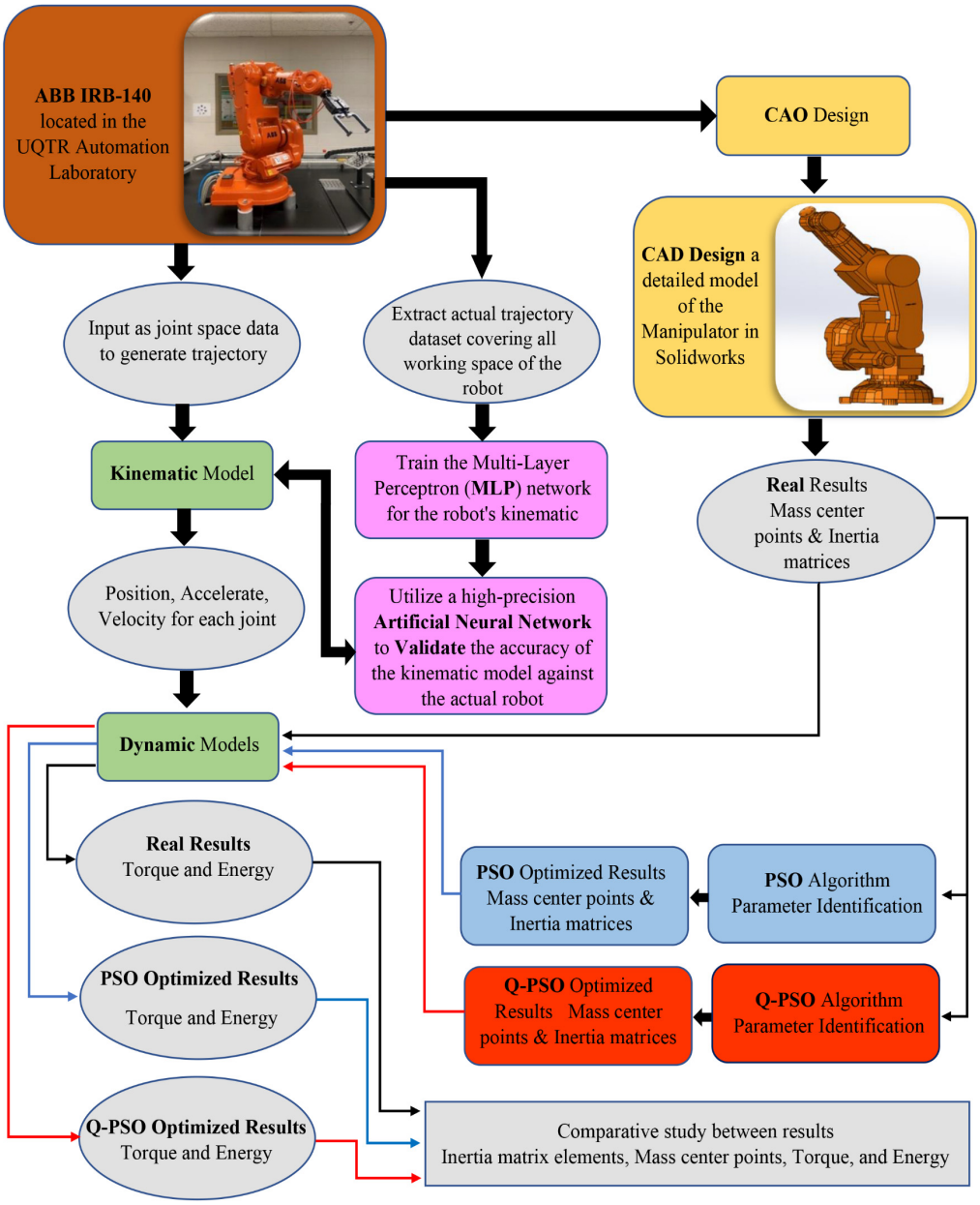


Figure 2. The flowchart for the Methodological Steps.

2.3. Kinematics of the Robot

The forward kinematics model determines the position and orientation of a robot's end-effector based on its joint angles and displacements relative to a reference frame, typically the base coordinate system [30,31]. This process depends on a global coordinate frame assigned to the robot base and local coordinate frames assigned to each joint. A typical

method for modeling forward kinematics is the Denavit–Hartenberg (D-H) convention, which utilizes a series of 4 × 4 homogeneous transformation matrices to describe the relative pose between consecutive joints [32]. Within this formalism, each joint pair is characterized by four parameters: the link length (a) and link twist (α), which remain constant, and the link offset (d) and joint angle (θ), which may vary depending on the joint type [33]. These parameters define the geometric relationships between links, enabling the precise determination of the end-effector's configuration. Figure 3 illustrates the allocation of D-H parameters and coordinate frames for a typical revolute joint [5].

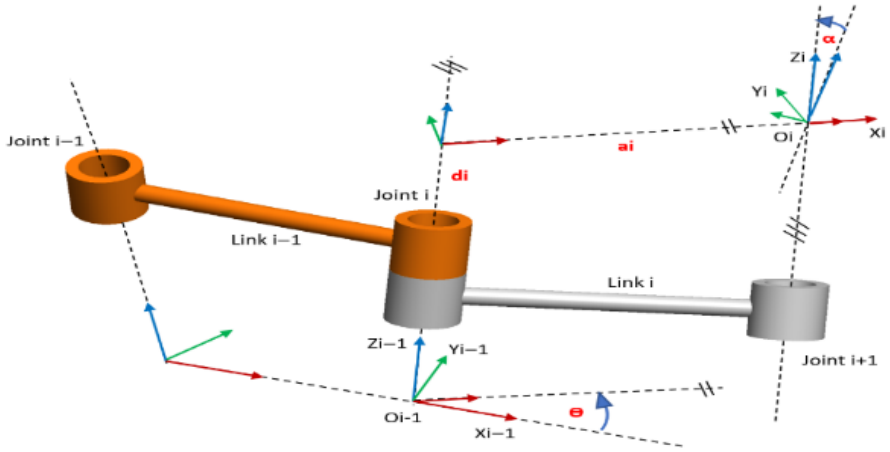


Figure 3. Rotational joint D-H parameters and link assignments [5].

The four transformations between the two axes can be defined as follows:

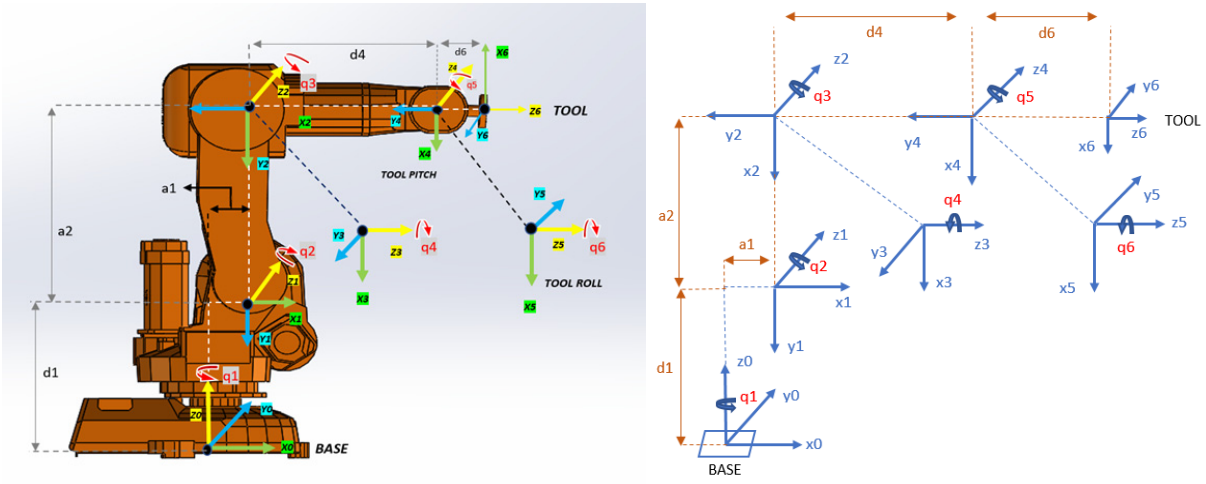
$$ {}^{i-1}_{i}T \;=\; Rot(x_{i\,-\,1},\, \alpha_{i\,-\,1})Trans(x_{i\,-\,1},\, a_{i\,-\,1})Rot(z_i\,,\, \theta_i\,)Trans(0,\, 0,\, d_i) \tag{1} $$

where ${}^{i-1}_{i}T$ is the homogeneous transformation matrix, Rot $(x_{i-1},\ \alpha_{i-1})$ is the rotation around an axis x_{i-1} by an angle α_{i-1}, Trans $(x_{i-1},\ a_{i-1})$ is the transfer along axis x_{i-1} to the value a_{i-1}, Rot $(z_i,\ \theta_i)$ is the rotation around axis z_i by an angle θ_i, and Trans $(0,\, 0,\, d_i)$ is the transfer along axis z to the value d.

Consequently, the corresponding homogeneous transformation matrix can be expressed as follows. In this matrix, C_i and S_i denote $\cos(\theta_i)$ and $\sin(\theta_i)$, respectively, where θ_i is the joint angle of the i-th joint. Similarly, C_{ij} and S_{ij} represent $\cos(\theta_i + \theta_j)$ and $\sin(\theta_i + \theta_j)$, respectively.

$$ {}^{i-1}_{i}T \;=\; \begin{pmatrix} C\theta_I & -SI & 0 & a_{i-1} \\ SI_i\, C\alpha_{i-1} & \mathrm{CI} & -S\alpha_{i-1} & -d_i S\alpha_{i-1} \\ S\theta_i S\alpha_{i-1} & C\theta_i\, S\alpha_{i-1} & C\alpha_{i-1} & d_i C\alpha_{i-1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2} $$

Figure 4 illustrates the assignment of coordinate frames to each joint of the ABB IRB 140 industrial robot, establishing the basis for its kinematic representation. Table 1 complements this figure by detailing the corresponding Denavit–Hartenberg (D-H) parameters, including link lengths, twists, offsets, and joint angles, which together define the spatial relationships between successive links. These frame assignments and kinematic parameters are configured with respect to a unified global coordinate framework, which is subsequently described to ensure geometric consistency across the robot's workspace and to facilitate accurate forward and inverse kinematic computations necessary for motion planning, control implementation, and simulation validation [7].



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

Figure 4. ABB IRB 140 frame assignments: (**a**) Frames represented on the real robot; (**b**) Frames symbolized using DH representation [7].

Table 1. Denavit–Hartenberg Parameters of the ABB IRB 140 [5,7].

Link	a (mm)	α (°)	d (mm)	q (°)
1	$a_1 = 70$	-90	$d_1 = 352$	q_1
2	$a_2 = -360$	0	0	$q_2 + 90$
3	0	-90	0	q_3
4	0	90	$d_4 = 380$	q_4
5	0	-90	0	q_5
6	0	0	$d_6 = 65$	q_6

Applying the Denavit–Hartenberg convention, each link's transformation matrix is obtained by substituting its specific parameters into relation (2). The sequential multiplication of these matrices yields the end-effector's pose concerning the base frame.

$$ {}^{0}_{6}T = {}^{0}_{1}T.\,{}^{1}_{2}T.\,{}^{2}_{3}T.\,{}^{3}_{4}T.\,{}^{4}_{5}T.\,{}^{5}_{6}T = \begin{pmatrix} r_{11} & r_{12} & r_{13} & X \\ r_{21} & r_{22} & r_{23} & Y \\ r_{31} & r_{32} & r_{33} & Z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3} $$

So that

$$ \begin{aligned} r_{11} &= -S_6(S_4C_1C_{2'3} + C_4S_1) - C_6(C_5(S_1S_4 - C_4C_1C_{2'3}) + S_5C_1C_{2'3}) \\ r_{12} &= S_6(C_5(S_1S_4 - C_4C_1C_{2'3}) + S_5C_1C_{2'3}) - C_6(S_4C_1C_{2'3} + C_4S_1) \\ r_{13} &= C_5C_1S_{2'3} - S_5(S_1S_4 - C_4C_1C_{2'3}) \\ r_{21} &= C_6(C_5(C_4S_1C_{2'3} + C_1S_4) - S_5S_1S_{2'3}) - S_6(S_4S_1C_{2'3} - C_1C_4) \\ r_{22} &= -S_6(C_5(C_4S_1C_{2'3} + C_1S_4) - S_5S_1S_{2'3}) - C_6(S_4S_1C_{2'3} - C_1C_4) \\ r_{23} &= C_5S_1S_{2'3} + S_5(C_4S_1C_{2'3} + C_1S_4) \\ r_{31} &= -C_6(S_5C_{2'3} + C_4C_5S_{2'3}) + S_4S_6S_{2'3} \\ r_{32} &= C_6S_4S_{2'3} + S_6(S_5C_{2'3} + C_4C_5S_{2'3}) \\ r_{33} &= C_5C_{2'3} - C_4S_5S_{2'3} \end{aligned} $$

The notation 2′ refers to the modified joint variable $q_2' = q_2 + \pi/2$, indicating an offset of $\pi/2$ applied to the original joint angle q_2. The X, Y, and Z position coordinates of the IRB140 robot relative to the base frame are computed as follows:

$$X = C_1 a_1 + d_6(C_5 C_1 S_{2'3} - S_5(S_1 S_4 - C_4 C_1 C_{2'3}) + d_4 C_1 S_{2'3} - C_1 C_2 a_2$$

$$Y = S_1 a_1 + d_6(C_5 S_1 S_{2'3} - S_5(C_4 S_1 C_{2'3} + C_1 S_4) + d_4 S_1 S_{2'3} - C_2 S_1 a_2$$

$$Z = d_1 + d_4 C_{2'3} + a_2 S_2 + d_6(C_5 C_{2'3} - C_4 S_5 S_{2'3})$$

2.4. Differential Kinematics of the Robot

Differential kinematics, a cornerstone of robotics, defines the relationship between the joints' angular velocities and the corresponding end-effector linear and angular velocities. The Jacobian matrix of the manipulators, which is an essential component for analyzing and controlling robotic motion and determining singularities and redundancy, can be derived [34].

Regarding the end-effector linear velocity vector $\dot{p}_e$, the angular velocity vector ω_e, and the joint velocity vector $\dot{q}$, J_p is the $(3 \times n)$ matrix that relates the linear velocity vector to the joint speed vector. J_o is the $(3 \times n)$ matrix that links the angular velocity vector to the joint speed vector is expressed in relations (4) and (5) or in the compact form represented in Equation (6) [35].

$$\dot{P}_e = J_p(q)\dot{q} \tag{4}$$

$$\omega_e = J_o(q)\dot{q} \tag{5}$$

$$V_e = \begin{pmatrix} \dot{P}_e \\ \omega_e \end{pmatrix} = J(q)\dot{q} \tag{6}$$

where V_e is the end-effector velocity and $\mathrm{J} = \begin{pmatrix} J_p \\ J_o \end{pmatrix}$ is the Jacobian matrix. To find the corresponding joint velocities for a desired end-effector position and orientation, the Jacobian matrix can be inverted, as represented in Equation (7) [7].

$$\dot{q} = J^{-1} V_e \tag{7}$$

where J^{-1} is the inverse of the Jacobian matrix.

2.5. Dynamic Model of the Robot

A dynamic model facilitates the representation of the robot's operation by accounting for joint acceleration forces and torque [36]. The Euler–Lagrange method is employed to define this dynamic model. The first three proximal joints of the ABB IRB 140 robot arm play a crucial role in determining the end-effector position and the mechanical loads [7].

The inertial parameters required for the dynamic model—specifically; the mass; the center of mass position; and the inertia tensor for each link—are extracted using CAD modeling software (SolidWorks Premium 2023). Assuming uniform density, the mass of each link is estimated based on its volume relative to the total volume of the robot. The position vectors of the mass centers are computed in the local coordinate system and then expressed in the base reference frame. The inertia matrices are initially defined with respect to the local link frames and are transformed into the base reference frame using appropriate rotation matrices:

$$I_i = R_i I_i^{\text{local}} R_i^T \tag{8}$$

Here, R_i is the rotation matrix from the local frame of link i to the base frame, and I_i^{local} is the inertia tensor at the center of mass of link i, computed in the local coordinate frame. The transformation ensures consistency in expressing all inertial parameters in a common reference frame.

The dynamic behavior of the robot is governed by the standard Euler–Lagrange formulation:

$$\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) \tag{9}$$

In this equation, τ is the vector of joint torques, $M(q)$ is the inertia matrix, $\ddot{q}$ is the vector of joint accelerations, $C(q,\dot{q})$ represents Coriolis and centrifugal effects, and $G(q)$ is the gravitational force vector.

The inertia matrix $M(q)$ is derived by summing both the translational and rotational kinetic energy contributions of each link, expressed as

$$M(q) = \sum_{i=1}^{3} \left[m_i J_{v_i}^T J_{v_i} + J_{\omega_i}^T I_i J_{\omega_i} \right] \tag{10}$$

In this expression, the first term $m_i J_{v_i}^T J_{v_i}$ accounts for the translational kinetic energy of link i based on the mass center's motion, while the second term $J_{\omega_i}^T I_i J_{\omega_i}$ represents the rotational kinetic energy using the inertia tensor transformed into the base frame.

The Jacobian matrix J_{v_i} maps joint velocities to the linear velocity of the center of mass of link i and is defined as

$$J_{v_i} = \begin{bmatrix} \vec{z}_0 \times \left(\vec{r}_{ci} - \vec{r}_0\right) & \vec{z}_1 \times \left(\vec{r}_{ci} - \vec{r}_1\right) & \vec{z}_2 \times \left(\vec{r}_{ci} - \vec{r}_2\right) \end{bmatrix} \tag{11}$$

Here, $\vec{z}_j$ is the unit vector along the axis of joint j, $\vec{r}_{ci}$ is the center of mass position of link i, and $\vec{r}_j$ is the position of joint j, all expressed in the base frame. This formulation captures how each joint's motion affects the linear velocity of the link's mass center.

The Jacobian matrix J_{ω_i} describes the angular velocity of link i as a function of joint velocities and is given by

$$J_{\omega_i} = \begin{bmatrix} \vec{z}_0 & \vec{z}_1 & \vec{z}_2 \end{bmatrix} \tag{12}$$

This angular velocity Jacobian assumes all three joints are revolute and contribute to the orientation of the link.

The gravitational torque vector $G(q)$ is computed by projecting the gravitational forces acting at each center of mass through their respective Jacobians:

$$G(q) = \sum_{i=1}^{3} m_i J_{v_i}^T g \tag{13}$$

Here, g is the gravitational acceleration vector, typically defined as $g = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^T$ in the base frame. This term accounts for the torque generated by the weight of each link acting at its center of mass.

The Coriolis and centrifugal term $C(q,\dot{q})$ are derived from the Christoffel symbols of the first kind:

$$C_{ijk} = \frac{1}{2}\left(\frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{jk}}{\partial q_i} \right) \tag{14}$$

Each element of the Coriolis matrix is then computed as

$$C_{ij} = \sum_{k=1}^{3} C_{ijk}\dot{q}_k \tag{15}$$

This formulation allows $C(q, \dot{q})\dot{q}$ to capture dynamic coupling effects between joints due to their relative velocities.

The complete dynamic model is not only useful for torque prediction but can also be rearranged to solve for joint accelerations, joint velocities, and positions given known torque inputs. This inverse dynamics form is written as

$$\begin{gathered} M(q)\ddot{q} = \tau - C(q, \dot{q}) - G(q) \\ \ddot{q} = M(q)^{-1}(\tau - C(q, \dot{q}) - G(q)) \\ \dot{q} = \int \ddot{q} dt + \dot{q}_0 \\ q = \int \dot{q} dt + q_0 \end{gathered} \tag{16}$$

As outlined in the modeling procedure, the mass center positions extracted from the CAD design are employed in the formulation of the Jacobian matrices J_{v_i}, which capture the translational motion of each link's center of mass, as well as in the derivation of the gravity vector $G(q)$. Concurrently, the inertia tensors obtained from the same CAD-based analysis, defined about the respective mass centers and transformed to the base frame, are integrated into the dynamic model through the term $J_{\omega i}^T I_i J_{\omega_i}$, which represents the rotational kinetic energy contribution within the overall inertia matrix $M(q)$.

2.6. CAD Designs of the Robot

A comprehensive CAD modeling of the real (Referenced) robot to obtain the mass properties of each link of the robot was performed. Mass parameter tools in SolidWorks (Premium 2023), Solid Edge ST4, and CATIA (via the DELMIA Catalogue of the robot) are used to ensure consistent units and high-precision geometry to authorize the accurate extraction of mass, center of mass, and inertia matrices with less than a 0.9% error. The results deliver the elements of the inertial matrix and the position of the mass center for each of the first three links, which are essential for formulating the dynamic model. It is important to highlight that the assumption of uniform density simplifies this calculation process. The CAD model illustrated in Figure 5 supports a high-resolution investigation of the multi-body dynamics and inter-component interactions of the case study. Table 2 compiles the derived mass property data for the proximal links [7].

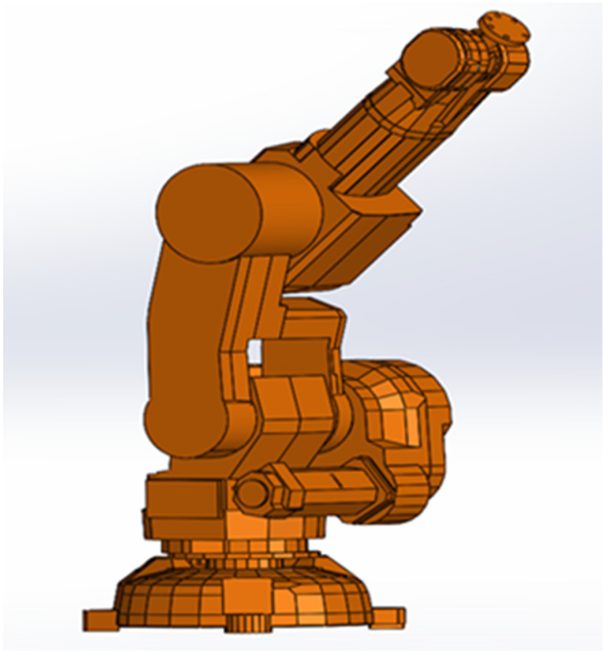


Figure 5. Detailed CAD model of the ABB IRB 140 robotic arm [7].

Table 2. Mass Property Results of the Robot Calculated by CAD Software.

ABB IRB 140	Parameters (unit)	Link 1	Link 2	Link 3
Mass parameters	*Weight* (kg)	35	25	18
	X_c (mm)	277.87	218.29	−24.56
	Y_c	373.12	229.73	−219.9
	Z_c	−199.03	112.43	−25.86
	I_{xx} (kg.m^2)	6.5	0.9	2.5
	I_{xy}	1.1	−0.03	−0.001
	I_{xz}	3.05	0.1	0.09
	I_{yy}	2.02	1.3	2.7
	I_{yz}	5.07	−0.01	−0.8
	I_{zz}	1.4	0.95	0.5

2.7. Particle Swarm Optimization Algorithm

Particle Swarm Optimization is a nature-inspired, population-based stochastic algorithm that emulates social behaviors observed in biological collectives such as bird flocks and fish schools [37,38]. Noted for its low algorithmic complexity, rapid convergence, and powerful global search capabilities, PSO has been widely applied to high-dimensional optimization problems [39,40].

Each particle represents a candidate solution within an n-dimensional decision space. Particles adapt their trajectories by iteratively updating their velocities and positions based on personal and collective experiences. The velocity update is expressed as [41]

$$\vec{v}_i(n+1) = \omega \vec{v}_i(n) + c_1 r_1 \left(\vec{p}_i - \vec{x}_i(n) \right) + c_2 r_2 \left(\vec{g} - \vec{x}_i(n) \right) \tag{17}$$

where ω is the inertia weight; c_1, c_2 are cognitive and social acceleration coefficients; and $r_1, r_2 \in U[0,1]$ are stochastic multipliers. The inertia weight is typically updated dynamically as

$$\omega = \omega_{max} - \left(\frac{\omega_{max} - \omega_{min}}{iter_{\ max}} \right) \times iter \tag{18}$$

The inertia weight ω is dynamically adjusted by decreasing linearly from $\omega_{\max}$ to $\omega_{\min}$ over the maximum number of iterations $iter_{max}$ to balance global exploration and local exploitation.

The updated position is

$$\vec{x}_i(n+1) = \vec{x}_i(n) + \vec{v}_i(n+1) \tag{19}$$

Particles are initially distributed randomly within the search space and are iteratively refined based on a problem-specific objective function that guides the swarm toward optimal solutions. The optimization process continues until either a predefined maximum number of iterations is reached or the solution meets a desired convergence threshold, typically expressed as an error tolerance criterion [37]. In the context of optimizing mass properties for the robotic arm, each particle represents a candidate solution comprising a set of inertia tensor elements and mass center coordinates, randomly initialized with associated velocities across a high-dimensional parameter space. These particles collectively explore the solution landscape by balancing exploration and exploitation through adaptive updates. The step-by-step procedure for applying the classical PSO algorithm to dynamic parameter identification is presented in Algorithm 1, outlining the initialization, evaluation, and update rules employed throughout the optimization process.

Algorithm 1: Classical Particle Swarm Optimization Steps

Notation:
N: Number of particles
MaxIter: Maximum number of iterations
x_{ij}^t: Position of particle i in dimension j at iteration t
$p_{best}{}_j^t$: Personal best angle of particle i
$g_{best}{}_j^t$: Global best angle in dimension j
v_{ij}^t: Velocity of particle i
ω: Inertia weight
c_1, c_2: Learning coefficients
r_1, r_2: Uniformly distributed random numbers in $(0, 1)$
f: Objective function
x^*: Optimal identified parameter set
Steps:
1: **Initialize** population of N particles with random positions x_{ij}^0 and velocities v_{ij}^0
2: **Evaluate** initial fitness $f\left(x_i^0\right)$ for all particles
3: **Set** $p_{best,\,i} \leftarrow x_i^0$, and identify $g_{best,\,j}$
4: **for** $t = 1$ to Maxiter **do**
5: **for** each particle i **do**
6: **Update** velocity:

$$v_{ij}^t = \omega v_{ij}^{t-1} + c_1 r_1 \left(p_{best,\,ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2 \left(g_{best,\,j}^{t-1} - x_{ij}^{t-1}\right)$$

7: **Update** position:

$$x_{ij}^t = x_{ij}^{t-1} + v_{ij}^t$$

8: **end for**
9: **Evaluate** fitness $f\left(x_i^t\right)$
10: **Update** $p_{best,\,i} \leftarrow x_i^t$ if improved
11: **Update** $g_{best,\,j}$ if improved
12: **Return** $x^* \leftarrow g_{best,\,j}$ as the optimal identified parameters

The objective function in the PSO algorithm minimizes the squared Euclidean norm between simulated dynamic parameters and reference values obtained from the CAD model. It ensures that the estimated mass center positions and inertia matrix elements are close to reference values.

$$f(o) = \sum_{i=1}^{N} \left\| p_{sim,\,i}(o) - p_{ref,\,i} \right\|^2 + \lambda \cdot \ Penalty\ (o) \tag{20}$$

Here, o is the candidate solution, $p_{sim,\,i}(o)$ are the simulated parameters of link i, and $p_{ref,\,i}$ are the corresponding CAD-based reference values and λ is the Regularization weight for the penalty term, which is adaptively updated as a function of the iteration index, progressively increasing to strengthen constraint enforcement as the optimization advances. The additional penalty term serves to enforce physical validity by penalizing candidate solutions that violate predefined parameter bounds, guiding the optimization process toward feasible and physically consistent solutions.

2.8. Quantum-Behaved Particle Swarm Optimization Algorithm

In the QPSO algorithm, each particle is represented using qubits, the fundamental units of quantum information, which can exist in a superposition of classical binary states. This contrasts with conventional binary or real-valued encodings. Mathematically, a single qubit is defined as [42–44]

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{21}$$

Here, α and β are complex numbers representing the probability amplitudes of the qubit's state that determine the possibility of the qubit collapsing to the respective basis states upon measurement. These amplitudes are subject to the normalization condition as below [45–47]:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{22}$$

In practical implementations of QPSO, the qubit is often expressed in terms of a quantum angle $\theta \in [0, \pi/2]$, yielding a real-valued representation [48]:

$$|\phi\rangle = sin\theta|0\rangle + cos\theta|1\rangle \tag{23}$$

Consequently, the amplitudes are defined as

$$\alpha = sin\theta,\ \beta = cos\theta \tag{24}$$

For a particle in an m-dimensional space, its state is defined by a vector of quantum angles. Each angle encodes a probability distribution over a binary state [47]:

$$\theta_i = [\theta_{i1}, \theta_{i2}, \ldots, \theta_{im}] \tag{25}$$

Each particle's angular velocity evolves according to both local and global attractors. The angular velocity is updated using the following equation [49,50]:

$$v_{ij}^{t+1} = \omega v_{ij}^t + c_1 r_1 \left[\theta_{ij}^t\,(p_{best}) - \theta_{ij}^t \right] + c_2 r_2 \left[\theta_{ij}^t\,(g_{best}) - \theta_{ij}^t \right] \tag{26}$$

where v_{ij}^t is the angular velocity of the i-th particle in the j-th dimension at iteration t, θ_{ij}^t is the current quantum angle, and $pbest_{ij}^t$ and $gbest_{ij}^t$ represent the personal and global best quantum angles, respectively. The quantum angle is then updated as follows [49]:

$$\theta_{ij}^{t+1} = \theta_{ij}^t + v_{ij}^{t+1} \tag{27}$$

Each quantum angle θ_j is measured by generating a uniformly distributed random number $r \in [0, 1]$ and comparing it to the squared amplitude $|\alpha_j|^2 = \cos^2(\theta_j)$. If $r \geq |\alpha_j|^2$, the bit is set to 1; otherwise, it is set to 0, as defined in relation (28):

$$x_j = \begin{cases} 1, & if\ r \geq |\alpha_j|^2 \\ 0, & if\ r < |\alpha_j|^2 \end{cases} \tag{28}$$

After each particle's quantum angle vector is updated according to relation (27), the corresponding quantum states are measured and collapsed into binary values based on the probabilistic formulation described in Equations (23) and (24). This collapse process enables the mapping of quantum representations into candidate solutions within the optimization space. These solutions are then evaluated using the objective function defined in relation (20), which assesses the fitness of each particle based on its estimated mass parameters. The complete sequence of operations involved in applying the Quantum-

behaved Particle Swarm Optimization (QPSO) algorithm to parameter identification is summarized in Algorithm 2.

Algorithm 2: Quantum-Behaved Particle Swarm Optimization Steps

Notation:
N: Number of particles
MaxIter: Maximum number of iterations
$\boldsymbol{\theta}_{\mathbf{ij}}^{\mathbf{t}}$: Current quantum angle of particle i in dimension j at iteration t
$\mathbf{p}_{\mathbf{best}\ \mathbf{j}}^{\mathbf{t}}$: Personal best angle of particle i
$\mathbf{g}_{\mathbf{best}\ \mathbf{j}}^{\mathbf{t}}$: Global best angle in dimension j
$\mathbf{v}_{\mathbf{ij}}^{\mathbf{t}}$: Angular velocity of particle i
$\boldsymbol{\omega}$: Inertia weight
$\mathbf{c}_1, \mathbf{c}_2$: Learning coefficients
$\mathbf{r}_1, \mathbf{r}_2$: Uniformly distributed random numbers in $(0, 1)$
f: objective function
$\boldsymbol{\theta}^*$: Optimal quantum angle solution
Steps:
1: **Initialize** population of N particles with quantum angles θ_{ij}^0
2: **Evaluate** initial fitness $f\left(\theta_i^0\right)$ for all particles
3: **Set** $p_{best}{}_{ij}^{0} \leftarrow \theta_{ij}^0$, and identify $g_{best}{}_{ij}^{0}$
4: **for** $t = 1$ to MaxIter do
5: **for** each particle i do
6: **Update** velocity:

$$v_{ij}^t = \omega v_{ij}^{t-1} + c_1 r_1 \left(pbest_{ij}^{t-1} - \theta_{ij}^{t-1} \right) + c_2 r_2 \left(gbest_j^{t-1} - \theta_{ij}^{t-1} \right)$$

7: **Update** position:

$$\theta_{ij}^t = \theta_{ij}^{t-1} + v_{ij}^t$$

8: **end for**
9: **Evaluate** new fitness $f\left(\theta_i^t\right)$
10: **Update** $p_{best}{}_{ij}^{t}$ if $f\left(\theta_i^t\right)$ improves
11: **Update** $g_{best}{}_{ij}^{t}$ if global best improves
12: **If** no improvement in g_{best} for a defined iteration, apply perturbation
13: **end for**
14: **Return** $\theta^* \leftarrow g_{best_j}$ as optimal solution set

2.9. Robot Performance Evaluation

To evaluate the impact of the optimal values derived from the mass parameters obtained through PSO and QPSO, a comprehensive and thorough analysis of the torque and energy consumption profiles is essential. This investigation is grounded in the dynamic model described in Equations (9) and (16). The energy consumption for each joint is formalized as follows:

$$E_i = \int_{t_0}^{t_j} \tau_i(t) \cdot \dot{q}_i(t) dt \tag{29}$$

Thus, E_i quantifies the total mechanical energy exerted by actuator i over the time interval $\left[t_0, t_j\right]$, serving as a critical metric for evaluating energy efficiency in robotic joint operation, trajectory planning, and overall control system performance.

3. Results and Discussion

The research begins by developing and validating the robot's kinematic and dynamic models using a high-precision MLP trained on real-world scenarios. Two optimization algorithms are then designed to explore optimized values for the mass parameters of the robot arm under identical conditions utilizing 120 particles over 150 iterations. Results were averaged over 30 independent optimization runs to ensure statistical reliability and minimize variation. The inertia weight was set to 0.99, and the acceleration coefficients c_1 and c_2 were set to 2.5 and 1.5, respectively, based on sensitivity-based tuning. Table 3 provides a comparative summary of the estimated mass properties of the robot arm, the mass center coordinates, and inertia matrix elements derived from the CAD-based reference model and those obtained through optimization using classical PSO and QPSO algorithms.

Table 3. Mass Parameters of Reference Model, PSO, and QPSO.

Links	Parameters	Reference Model	QPSO	PSO
Link 1	*Xc* (m)	0.27787	0.27939	0.28513
	Yc	0.37312	0.3742	0.37757
	Zc	−0.19903	−0.19905	−0.205
	Ixx (Kg.m^2)	6.5	6.5236	6.5701
	Ixy	1.1	1.1101	1.0255
	Ixz	3.05	3.0539	3.1819
	Iyy	2.02	2.056	2.0662
	Iyz	5.07	5.1022	5.1781
	Izz	1.4	1.4127	1.2907
Link 2	*Xc* (m)	0.21829	0.21953	0.23221
	Yc	0.22973	0.23236	0.23766
	Zc	0.11243	0.11284	0.10949
	Ixx (Kg.m^2)	0.9	0.90239	0.84598
	Ixy	−0.03	−0.03039	−0.02873
	Ixz	0.1	0.10049	0.099257
	Iyy	1.3	1.3105	1.2213
	Iyz	−0.01	−0.01000	−0.01079
	Izz	0.95	0.9605	0.9994
Link 3	*Xc* (m)	−0.02456	−0.02462	−0.02581
	Yc	−0.21996	−0.22108	−0.21592
	Zc	−0.02586	−0.02593	−0.02729
	Ixx (Kg.m^2)	2.5	2.5213	2.5711
	Ixy	−0.001	−0.00100	−0.00092
	Ixz	0.09	0.09162	0.09525
	Iyy	2.7	2.7082	2.7503
	Iyz	−0.8	−0.80558	−0.75339
	Izz	0.5	0.50243	0.49506

Figure 6 compares the referenced and optimized inertia matrix elements using PSO and QPSO. While both methods yield effective refinement, QPSO proves enhanced sensitivity, particularly in off-diagonal terms, capturing inertial coupling effects more accurately and achieving a more physically coherent representation of mass distribution and inter-axis dynamics relative to the reference model.

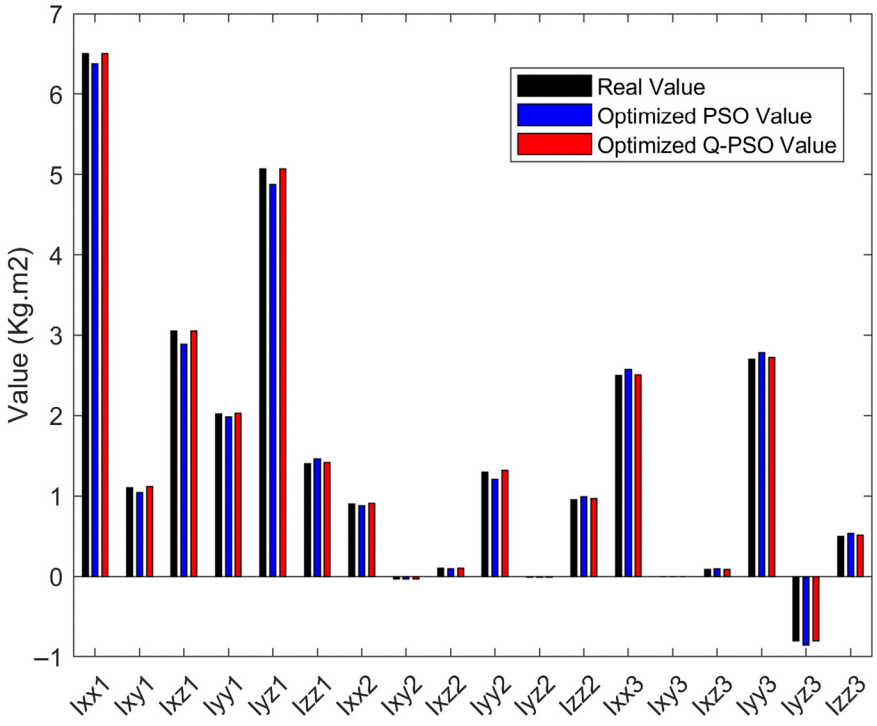


Figure 6. Bar chart of Reference vs. Optimized Inertia Matrix Elements.

Figure 7 illustrates that both algorithms effectively explore around the reference model's mass center points. However, QPSO achieves closer alignment with reference values, showing better global search and reduced risk of premature convergence.

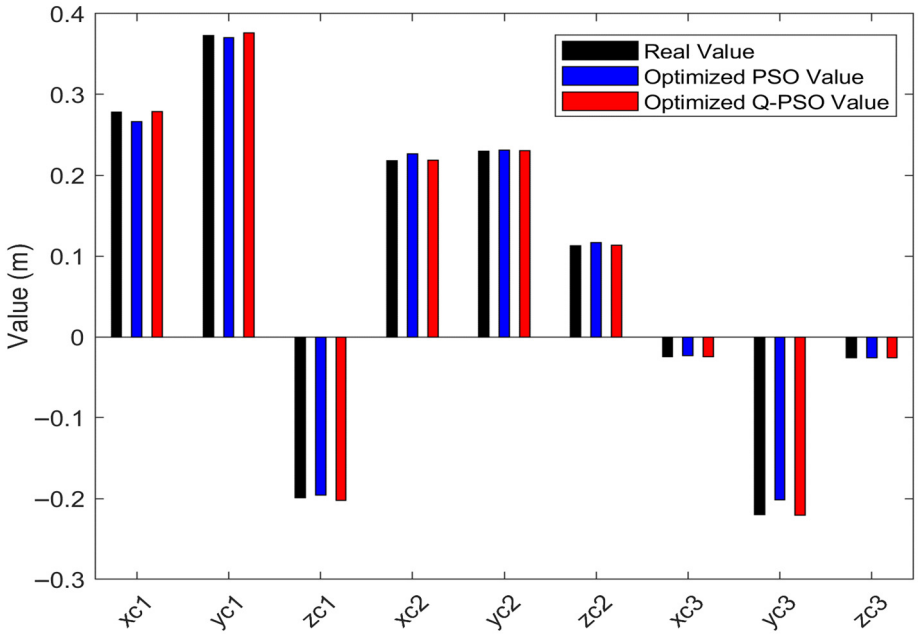


Figure 7. Bar chart of Reference vs. Optimized Mass centers position.

To quantify the accuracy of the identified parameters, the Mean Absolute Percentage Error (MAPE) was computed separately for the mass center coordinates and inertia matrix elements using the following formulation:

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{p_i^{est} - p_i^{ref}}{p_i^{ref}}\right| \times 100 \tag{30}$$

So that, p_i^{est} represents the ith parameter estimated by the algorithms, p_i^{ref} is the corresponding reference value, and N denotes the total number of parameters within the group 9 for mass center coordinates and 18 for inertia matrix.

Compared to the CAD-based reference values, QPSO achieved a MAPE of 0.43% for mass center coordinates and 0.76% for inertia tensor elements. In contrast, PSO displayed notably larger errors of 2.94% and 4.66%, respectively. These results prove QPSOs better precision in approximation.

Based on Figure 8, both algorithms show a rapid initial descent. PSO prematurely plateaus after approximately 27 iterations, indicating potential entrapment in local minima. In contrast, QPSO shows a more gradual yet consistent convergence toward lower fitness values, revealing its effective global search capability and long-term accuracy.

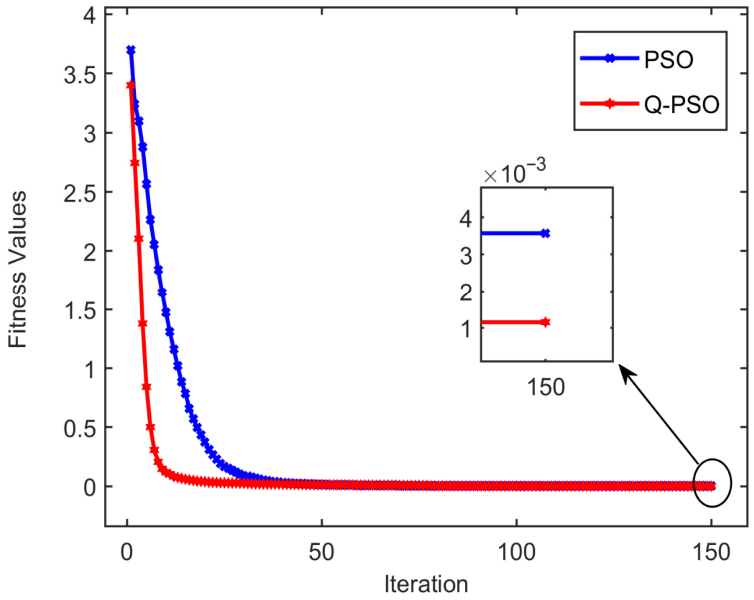


Figure 8. Convergence behavior of PSO and QPSO algorithms.

Figure 9 presents the normalized distributions of the final Global Best Scores achieved through 30 independent optimization runs for both the classical PSO and Quantum-behaved PSO algorithms. The boxplots indicate that QPSO consistently outperforms PSO in terms of solution stability and convergence accuracy. Notably, QPSO exhibits a lower median fitness value, a narrower interquartile range, and a decrease in the number of extreme values and outliers. Furthermore, the mean fitness value for QPSO was approximately 1.02, which is significantly lower than the 2.09 observed in PSO, thereby confirming the superior performance of the quantum-behaved variant in minimizing the objective function. While QPSO exhibits a slightly higher standard deviation (0.923 compared to 0.901), this can likely be attributed to its broader exploration behavior, which positively influences its ability to escape local minima and converge towards globally optimal solutions. This advantage is especially beneficial in high-dimensional, nonlinear search spaces,

such as robotic dynamic parameter identification, where the landscape is highly complex and traditional methods often face challenges with stagnation.

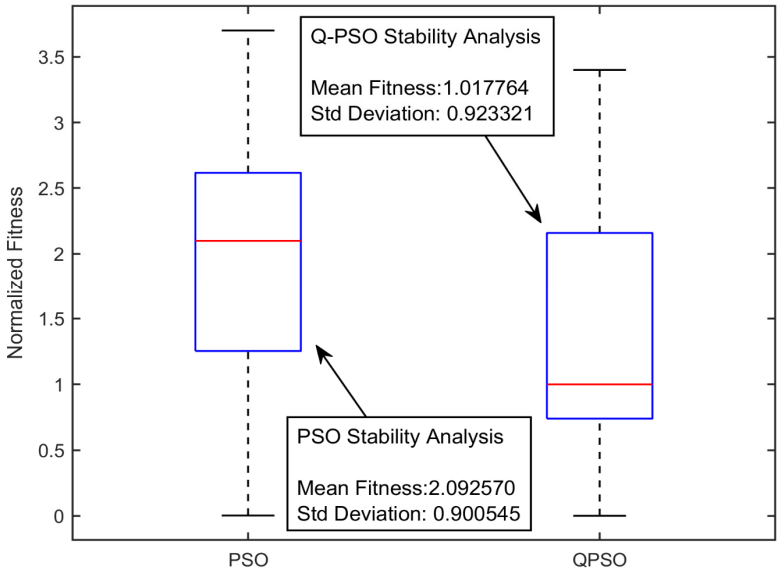


Figure 9. Final fitness values for PSO and QPSO over 30 runs.

In summary, on the same computational platform (Intel® Core™ *i7CPU@3.2GHz*, 32 *GB* RAM, Windows 10, MATLAB R2024a), QPSO achieved convergence in 32 ± 3 iterations on average across 30 runs, whereas PSO required 47 ± 4 iterations. This corresponds to approximately 7.9% less time per iteration and an overall convergence time reduction of about 32%. Furthermore, the mean final fitness value obtained by QPSO was approximately 51.35% lower than that of PSO, confirming both faster convergence and superior solution quality. These results affirm the robustness of QPSO in overcoming stochastic initialization effects and maintaining consistency across multiple runs while offering improved generalization.

To establish the accurate model of the dynamic model, the Kinematic model of the robot was validated by using a high-precision MLP neural network architecture [28,29], which was employed and trained on over 40,000 samples from the ABB IRB140 robot, which was actively engaged in executing physical tasks, as depicted in Figure 10, equipped with a singularity avoidance setup.

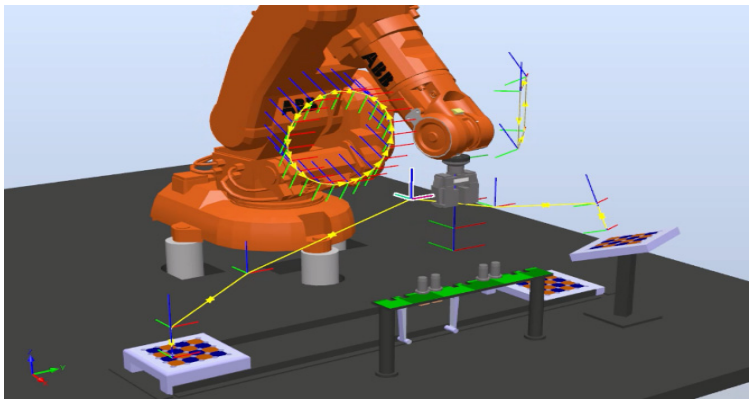


Figure 10. Robot arm during execution of assigned tasks [28,29].

The network architecture, shown in Figure 11, consisted of four hidden layers with 488 neurons. The classical MLP architecture utilized for kinematic validation was meticulously designed, with hyperparameters selected through preliminary experimentation and informed by best practices in function approximation tasks. A single hidden layer comprising 50 neurons was implemented to strike a balance between representational power and computational efficiency. Sigmoid activation functions were chosen for their established effectiveness in low-dimensional trajectory modeling. Input features, such as Cartesian positions and velocities, were normalized to a range of $[-1, 1]$ to improve convergence stability. The model was trained using a standard backpropagation algorithm with an adaptive learning rate, initially set at 0.01 and gradually decayed over the epochs. This configuration promoted smooth and consistent learning while helping to avoid local minima and overfitting. The model's convergence was evaluated by minimizing the Mean Squared Error (MSE), with training ceasing once the validation loss plateaued [28,29].

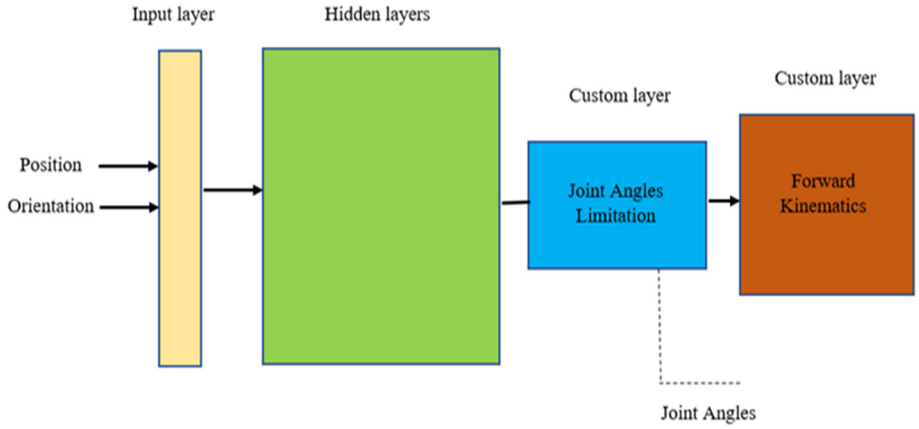


Figure 11. Robot's Arm MLP Neural Network Architecture [28,29].

To evaluate the performance of the artificial neural network (ANN) in predicting the end-effector's position, a standard Euclidean distance metric was employed. This metric quantifies the deviation between the predicted and target positions of the end-effector in 3D Cartesian space. The position error d is computed as [28,29]

$$d = \sqrt{(X - X')^2 + (Y - Y')^2 + (Z - Z')^2} \tag{31}$$

where $P = [X, Y, Z]$ represents the predicted position vector of the end-effector, and $P' = [X', Y', Z']$ denotes the desired or reference position. This scalar value d serves as an intuitive and geometric measure of the ANNs spatial accuracy, enabling a clear comparison between the actual and estimated kinematic behavior.

To ensure stable motion and prevent singularity-related issues in the robot's workspace, a regularization strategy based on the Jacobian matrix's condition number was employed. Singularities typically arise when the Jacobian matrix J becomes ill-conditioned or non-invertible, especially when $\det(J) = 0$ or the matrix approaches rank deficiency. These conditions can lead to excessive joint velocities or torque spikes, destabilizing the motion control process. To address this, the proximity to singularities was quantified using the Jacobian's condition number, defined as

$$\kappa(J) = \|J\| \cdot \|J^{-1}\| \tag{32}$$

where $\|J\|$ is the norm of the Jacobian matrix and $\|J^{-1}\|$ is the norm of its inverse, A high value of $\kappa(J)$ indicates greater sensitivity to input changes and signals a configuration near

singularity. A threshold of $\kappa > 1000$ was adopted to identify near-singular zones during the training phase.

To avoid such regions, the neural network's loss function was modified as

$$Loss\ Function = MAE\left(P_{Target}, P_{Calc}\right) + f(Activation\ Function, \kappa(Jacobian)) \quad (33)$$

In this formulation, P_{Target} is the desired position of the end-effector, P_{Calc} is the predicted position generated by the neural network, MAE represents the mean absolute error between them, $\kappa(Jacobian)$ quantifies how close the configuration is to a singular point based on the Jacobian's sensitivity, and f is a penalizing function influenced by the neural network's activation dynamics and the condition number value [28,29].

The network achieved a final training position error of 0.48 mm and a validation error of 0.61 mm, as illustrated in Figure 12, confirming its effectiveness in supporting high-accuracy kinematic validation. These results underscore the reliability of the neural network in modeling nonlinear kinematic relationships, providing a solid foundation for subsequent dynamic analysis and control integration.

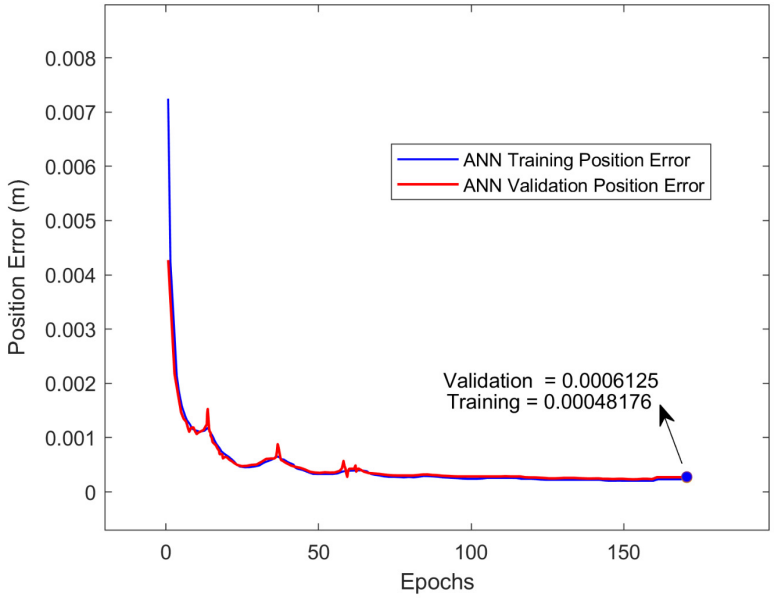


Figure 12. Position Error of MLP Neural Network.

Following the successful validation of the robot's kinematic model utilizing a high-precision multilayer perceptron network, three distinct dynamic models were established. These models incorporated mass property values derived from CAD-based reference data, classical Particle Swarm Optimization, and Quantum-behaved PSO algorithms, respectively. This modeling phase aimed to evaluate how different parameter estimation methods influenced the overall performance and dynamic behavior of the robotic system.

To conduct this comparative analysis, a robust Sliding Mode Control (SMC) strategy was devised and implemented, as depicted in Figure 13 [30]. This controller was integrated into a Simulink environment, operating in conjunction with the validated kinematic and dynamic models. The SMC architecture featured real-time trajectory tracking, torque feedback, and external disturbance modules, facilitating the evaluation of model robustness under both nominal and perturbed conditions.

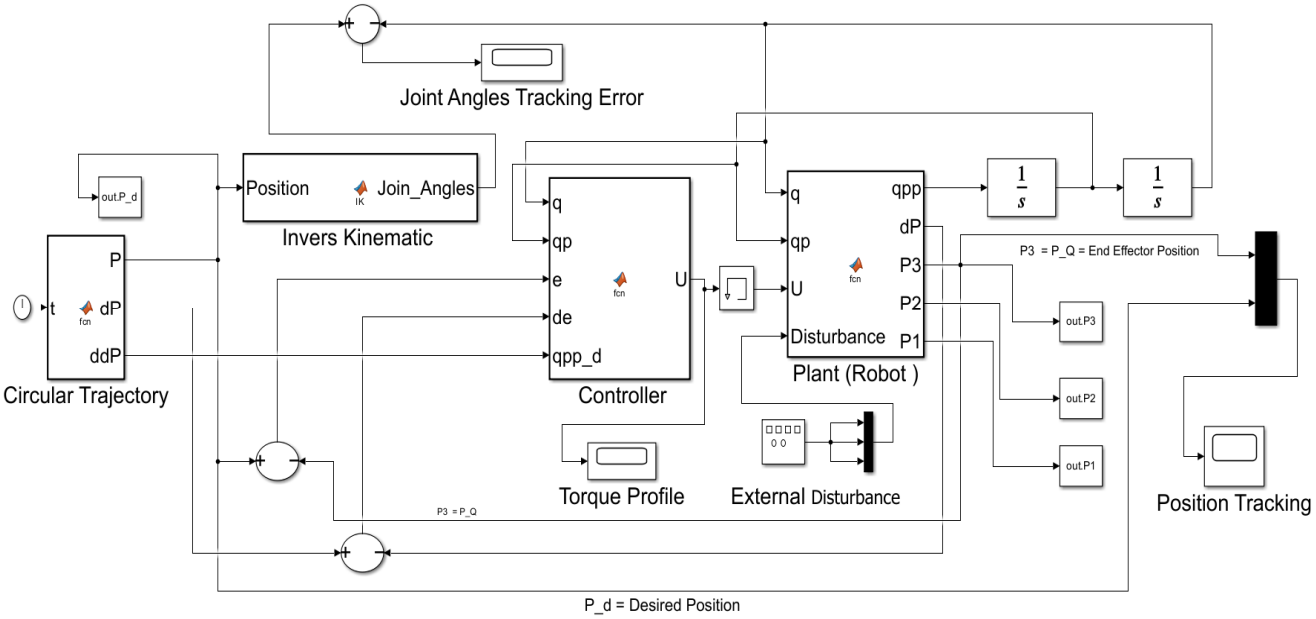


Figure 13. Simulink block diagram of the SMC-based circular trajectory [30].

A circular end-effector trajectory was predefined as the reference path for testing, executed over a simulation period of 30 s. As illustrated in Figure 14, the 3D visualization confirms that the robot effectively followed the desired circular trajectory within its operational workspace. This setup allowed for a precise assessment of each dynamic model's capability to respond to control inputs and maintain tracking accuracy over time.

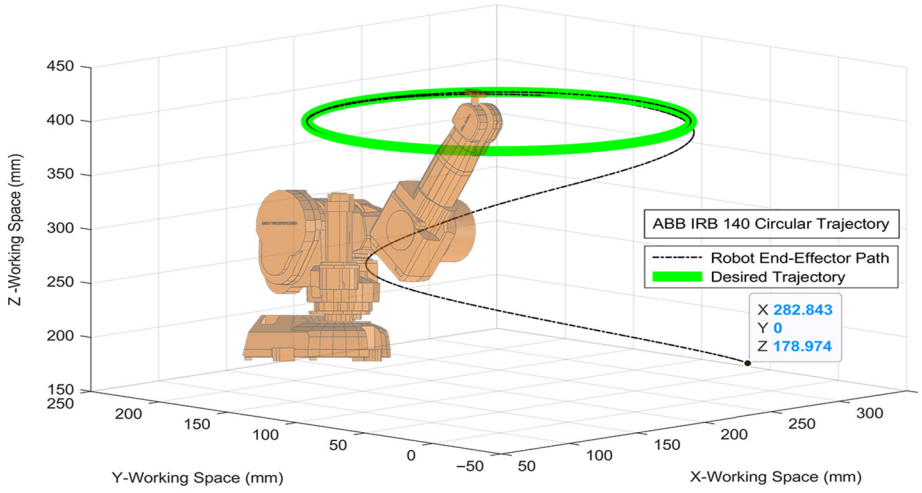


Figure 14. The 3-D visualization of the robot executing circular task.

The comparative performance analysis of the three models, CAD-based as the referenced model, PSO-optimized, and QPSO-optimized, was carried out by examining controller response characteristics, torque profiles, and energy consumption under identical trajectory planning tasks. This comprehensive integration of control and modeling under realistic task constraints provided valuable insights into the practical effectiveness of the proposed optimization strategies.

Controller feedback responses, torque, and energy metrics for the three models were considered nominal and disturbed conditions involving a 40% amplitude sinusoidal perturbation. To ensure statistical reliability and minimize variation, all reported results represent an average of over 30 independent simulation runs.

The results in Table 4 evaluate the tracking accuracy, controller responsiveness, and stability of Arm 3 of the robot, which represents the position of the end-effector under both nominal and disturbed conditions. In the nominal case, the PSO model excels with the fastest rise and settling times, minimal overshoot, and low steady-state error, making it ideal for precise, rapid trajectory tracking.

Table 4. Performance of Angular Position for the Different Models Under Disturbance.

Performances	Rising Time (s)	Settling Time (s)	Overshoot (%)	Steady State Error (%)
Arm 3 (End-effector)—Nominal scenario				
Real Model	0.577	0.599	1.85	0.0899
PSO Model	0.462	0.499	1.69	0.0607
QPSO Model	0.515	0.531	1.77	0.0799
Arm 3 (End-effector)—Disturbed scenario				
Real Model	0.654	0.791	1.99	0.0985
PSO Model	0.589	0.699	1.91	0.0851
QPSO Model	0.613	0.713	1.92	0.0910

The QPSO model, while slightly more conservative with higher overshoot, maintains competitive timing and closely mirrors the referenced model. Under disturbances, all models show performance decline, but the PSO model retains the fastest response. However, the QPSO model is more stable, robust, and reasonable at resisting perturbations and handling uncertainties.

The torque profiles observed under nominal and disturbed conditions, as illustrated in Figures 15 and 16, reveal differences in model performance. The QPSO model shows a close alignment with the reference model and also smoother transitions, minimal oscillations, and effective torque regulation, which suggests practical parameter identification and robust control. In contrast, the PSO model exhibits greater variability and more oscillations.

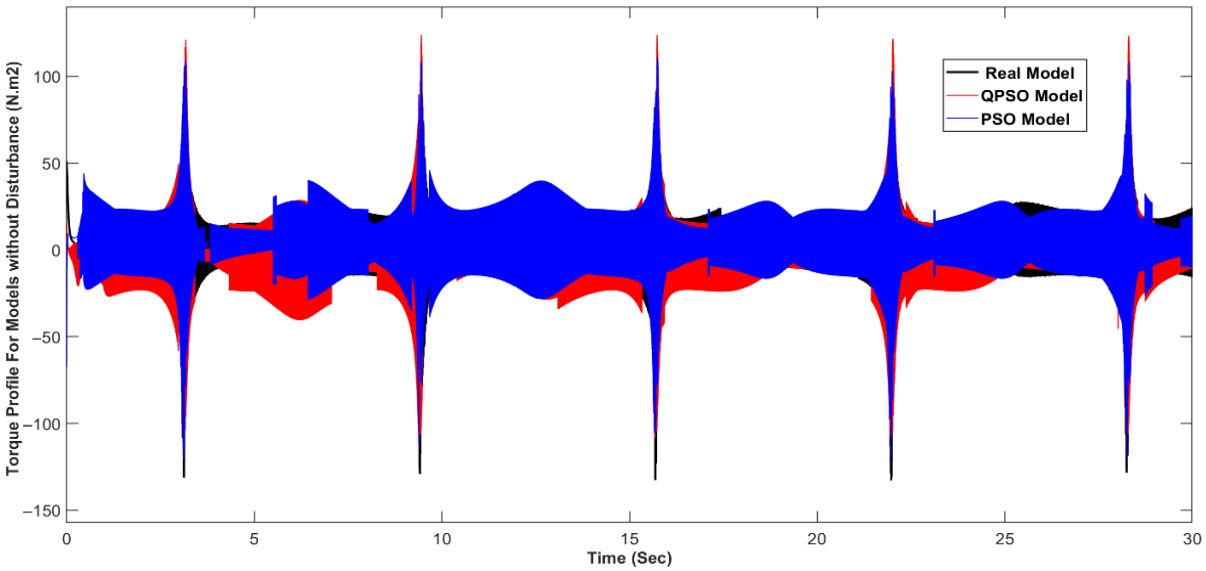


Figure 15. Torque profiles comparison under nominal conditions for three dynamic models.

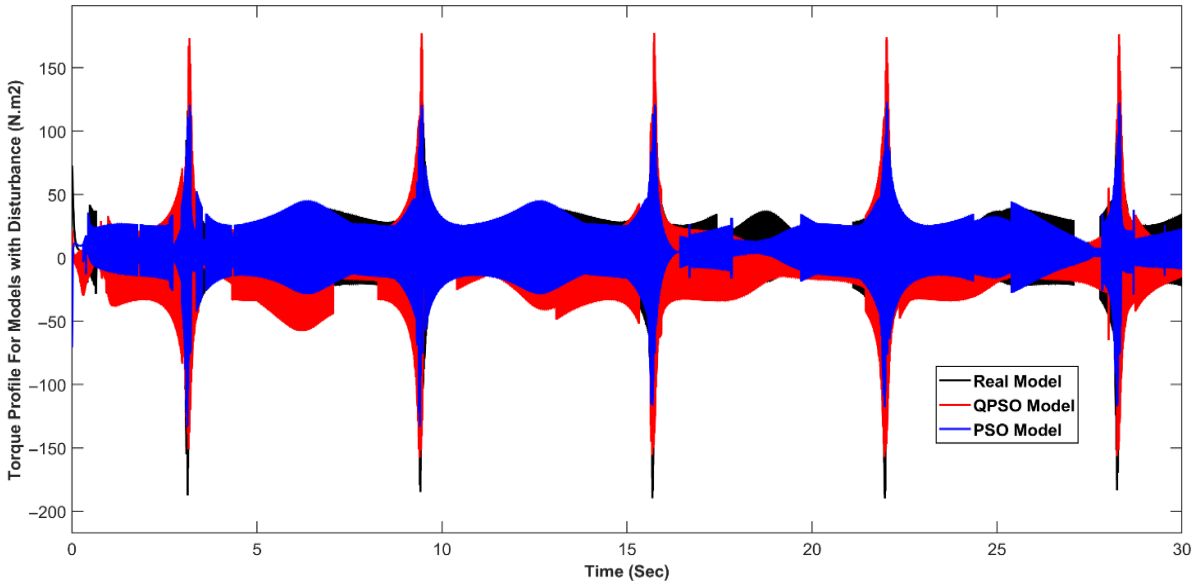


Figure 16. Torque profiles comparison under disturbed conditions for three dynamic models.

Under a 40% sinusoidal disturbance, the QPSO model demonstrates torque behavior that closely corresponds to that of the reference model. In contrast, the PSO model exhibits increased fluctuations, highlighting its susceptibility to disturbances. All models react to the disturbance, with the torque profile being affected and the torque demand increasing to control the robot and maintain the desired trajectory. However, QPSO still reveals better stability, characterized by smoother transitions, minimal oscillations, and adequate torque regulation.

The energy consumption results in Figure 17 compare the behavior of the model and the impact of three models on the energy profile of the robot during motion planning. All models exhibit a rising energy trend, consistent with the expected robotic behavior. The QPSO model, which closely aligns with the Real (Referenced) model's energy profile, provides a reassuring level of accuracy, indicating minimal actuator workload deviation. In contrast, the PSO model exhibits higher initial consumption and increasing divergence, indicating greater controller effort.

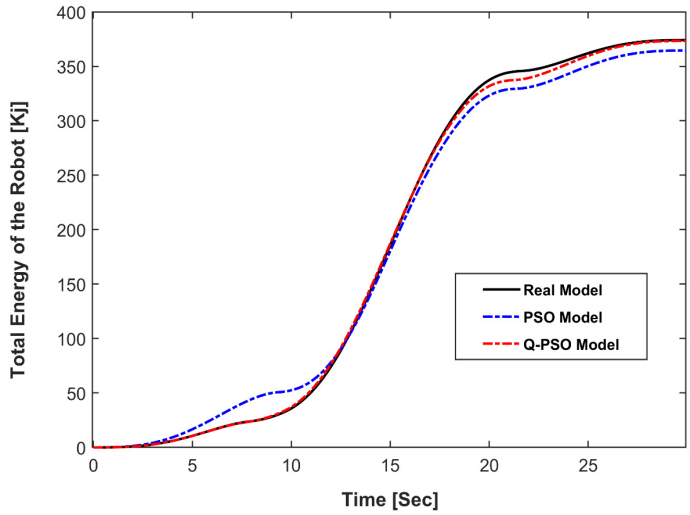


Figure 17. Energy consumption profiles for the three models.

The QPSO algorithm showed competitive performance compared to classical PSO, exhibiting a closer alignment with SolidWorks-derived ground truth parameters. This resulted in smoother torque profiles and practical control precision. Moreover, QPSO captured dynamic behaviors that reflected actual energy consumption patterns while maintaining robustness against disturbances. Table 5 provides a comparative overview of the QPSO and PSO algorithms across all evaluated performance dimensions.

Table 5. Comparative Overview of QPSO and PSO Performance Across Key Evaluations.

Evaluation Criterion	QPSO Performance	PSO Performance	Supporting
Parameter Estimation Accuracy	Higher estimation fidelity	larger deviations	Figures 6 and 7
Convergence Behavior	Smooth, stable convergence	Rapid descent—early plateau	Figure 8
Statistical Robustness (30 runs)	Lower median fitness	Higher variance—outlier spread	Figure 9
Trajectory Tracking (Nominal)	Comparable accuracy	Slightly faster	Table 4
Trajectory Tracking (Disturbed)	More resilient to perturbations	Some degradation under noise	Table 4
Torque Profile (Nominal)	Smoother torque transitions	Higher variability	Figure 15
Torque Profile (Disturbed)	Maintains regulated torque	Greater instability	Figure 16
Energy Consumption Trend	Closely follows reference model	Higher demand	Figure 17
Overall Control Robustness	Consistent behavior	More sensitive to noise	Figures 14–17

It is worth noting that the objective function was intentionally formulated to minimize deviations from CAD-based reference parameters. Alternative formulations of objective functions could directly target performance indices such as torque, energy, or trajectory error. The parameter-centric approach provided a reliable foundation for reconstructing the physical model. Extending this work to include performance-driven objective functions remains a promising direction for future research.

4. Conclusions

Current research presented a comparative investigation of classical Particle Swarm Optimization and Quantum-behaved PSO for identifying the dynamic parameters of a six-degree-of-freedom industrial robotic arm. The primary objective was to estimate the center of mass and the elements of the inertia matrix of the robot links, which are essential for accurate simulation, energy-efficient operation, and advanced control.

Practical results confirmed that both PSO and QPSO effectively explored the parameter space of mass parameters. However, QPSO exhibited better alignment with the SolidWorks-derived reference model, generating lower mean absolute percentage errors of 0.76% and 0.43%. The QPSO-based dynamic model enabled smoother torque transitions, improved energy consistency, and delivered more stable control behavior in the presence of external disturbances, highlighting its robustness in nonlinear, high-dimensional robotic environments.

The study presents a multi-layered approach to identifying the mass properties of industrial robot arms based on physically accurate data directly extracted from actual robots and validated by CAD design tools. To ensure practical validation, a kinematic model was first verified using a high-precision MLP trained on actual joint motion data. The study then analyzes and compares the robot's dynamic behavior performance by implementing the explored values for mass parameters using PSO and QPSO algorithms, incorporating a robust SMC, and evaluating under identical trajectory planning tasks with and without external disturbances.

While the proposed method presented promising outcomes, several limitations must be acknowledged. Accurate estimation of link masses, centers of mass, and inertia tensors remains a challenging task, as such data are often proprietary and not publicly disclosed

by robot manufacturers. In this study, these parameters were derived from a high-fidelity CAD model under simplifying assumptions, such as homogeneous material properties and uniform density, which may introduce minor deviations from the actual physical robot. The work was conducted exclusively on a 6-DOF rigid industrial manipulator following a pre-defined circular trajectory, which may limit the generalizability of the findings to robots with different kinematic structures, higher degrees of freedom, flexible links, or redundant configurations, as well as to other motion profiles. Furthermore, the current implementation was performed in an offline simulation environment and did not explicitly address real-time constraints, sensor noise, or unmodeled uncertainties encountered in embedded robotic systems.

Future work will focus on extending the methodology to additional robotic platforms to assess scalability and on implementing a modular modeling architecture that facilitates adaptation to diverse robot geometries, kinematic chains, and dynamic properties. In addition, future developments will explore control-aware objective functions that incorporate trajectory tracking performance, torque smoothness, and energy efficiency into the optimization process. Efforts will also be directed toward hardware implementation using embedded platforms to enable real-time evaluation and closed-loop experimentation. Finally, enhancements to the QPSO algorithm, such as adaptive attractor dynamics, noise-resilient architectures, and self-tuning mechanisms, are envisioned to improve robustness and generalization across various robotic applications.

Author Contributions: Conceptualization, M.F. and N.Z.; Methodology, M.F. and N.Z.; Software, M.F.; Validation, N.Z.; Formal analysis, M.F.; Investigation, M.F.; Resources, N.Z.; Data curation, N.Z.; Writing – original draft, M.F.; Writing – review & editing, N.Z.; Visualization, M.F.; Supervision, N.Z.; Project administration, N.Z.; Funding acquisition, N.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was financially supported by NSERC Canada.

Data Availability Statement: The data supporting the findings of this study are not publicly available, as they are part of ongoing research and future work. Data may be shared upon reasonable request after the completion of future studies.

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

Abbreviations

The following abbreviations are used in this manuscript:

CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAO	Computer-Aided Optimization
DE	Differential Evolution
DOF	Degrees of Freedom
EKF	Extended Kalman Filter
GWO	Grey Wolf Optimizer
KF	Kalman Filter
LS	Least Squares
MAPE	Mean Absolute Percentage Error
MLP	Multilayer Perceptron
PSO	Particle Swarm Optimization
QPSO	Quantum-behaved Particle Swarm Optimization
RLS	Recursive Least Squares
SMC	Sliding Mode Control

References

1. Urrea, C.; Pascal, J. Design and validation of a dynamic parameter identification model for industrial manipulator robots. *Arch. Appl. Mech.* **2021**, *91*, 1981–2007. [CrossRef]
2. Memar, A.H.; Esfahani, E.T. Modeling and dynamic parameter identification of the schunk powerball robotic arm. in International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. *Am. Soc. Mech. Eng.* **2015**, *5C*, V05CT08A024.
3. Urrea, C.; Pascal, J. Design, simulation, comparison and evaluation of parameter identification methods for an industrial robot. *Comput. Electr. Eng.* **2018**, *67*, 791–806. [CrossRef]
4. Duan, S.; Shi, L.; Wang, L.; Liu, G. An uncertainty inversion technique using two-way neural network for parameter identification of robot arms. *Inverse Probl. Sci. Eng.* **2021**, *29*, 3279–3304. [CrossRef]
5. Fazilat, M. *Modélisation Avancée* et Commande d'un Bras Robot Manipulateur Industriel = Advanced Modelling and Control of Industrial Robotic Manipulator Arm. Master's Thesis, Université du Québec à Trois-Rivières, Trois-Rivières, QC, Canada, 2022. Available online: https://depot-e.uqtr.ca/id/eprint/10597 (accessed on 14 August 2025).
6. Azizkhani, M.; Godage, I.S.; Chen, Y. Dynamic control of soft robotic arm: A simulation study. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3584–3591. [CrossRef]
7. Fazilat, M.; Zioui, N. The impact of simplifications of the dynamic model on the motion of a six-jointed industrial articulated robotic arm movement. *J. Robot. Control* **2024**, *5*, 173–186. [CrossRef]
8. Xu, W.; Hu, Z.; Zhang, Y.; Liang, B. On-orbit identifying the inertia parameters of space robotic systems using simple equivalent dynamics. *Acta Astronaut.* **2017**, *132*, 131–142. [CrossRef]
9. Bingül, Z.; Karahan, O. Dynamic identification of Staubli RX-60 robot using PSO and LS methods. *Expert Syst. Appl.* **2011**, *38*, 4136–4149. [CrossRef]
10. Zhong, F.; Liu, G.; Lu, Z.; Hu, L.; Han, Y.; Xiao, Y.; Zhang, X. Dynamic parameter identification based on improved particle swarm optimization and comprehensive excitation trajectory for 6R robotic arm. *Ind. Robot* **2024**, *51*, 148–166. [CrossRef]
11. Leboutet, Q.; Roux, J.; Janot, A.; Guadarrama-Olvera, J.R.; Cheng, G. Inertial parameter identification in robotics: A survey. *Appl. Sci.* **2021**, *11*, 4303. [CrossRef]
12. Vaze, R.; Deshmukh, N.; Kumar, R.; Saxena, A. Development and application of quantum entanglement inspired particle swarm optimization. *Knowl. Based Syst.* **2021**, *219*, 106859. [CrossRef]
13. Baş, E. Improved particle swarm optimization on based quantum behaved framework for big data optimization. *Neural Process. Lett.* **2023**, *55*, 2551–2586. [CrossRef]
14. Cao, H.Q.; Nguyen, H.X.; Nguyen, T.T.; Nguyen, V.Q.; Jeon, J.W. Robot calibration method based on extended Kalman filter–dual quantum behaved particle swarm optimization and adaptive neuro-fuzzy inference system. *IEEE Access* **2021**, *9*, 132558–132568. [CrossRef]
15. Dereli, S.; Köker, R. A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: Quantum behaved particle swarm algorithm. *Artif. Intell. Rev.* **2020**, *53*, 949–964. [CrossRef]
16. Ma, W.; Du, Q.; Zhu, R.; Han, W.; Chen, D.; Geng, Y. Research on inverse kinematics of redundant robotic arms based on flexibility index. *IEEE Robot. Autom. Lett.* **2024**, *9*, 7262–7269. [CrossRef]
17. Numbi, J.; Zioui, N.; Tadjine, M. Quantum Particle Swarm Optimisation Proportional-Derivative Control for Trajectory Tracking of a Car-like Mobile Robot. *Electronics* **2025**, *14*, 832. [CrossRef]
18. Yan, D.; Lu, Y.; Levy, D.; Du, W.-B. Parameter identification of robot manipulators: A heuristic particle swarm search approach. *PLoS ONE* **2015**, *10*, e0129157. [CrossRef]
19. Tang, M.; Yan, Y.; An, B.; Wang, W.; Zhang, Y. Dynamic parameter identification of collaborative robot based on WLS-RWPSO algorithm. *Machines* **2023**, *11*, 316. [CrossRef]
20. Ayinalem, Z.A.; Kassie, A.T.; Bianchi, D. PSO Tuned Super-Twisting Sliding Mode Controller for Trajectory Tracking Control of an Articulated Robot. *J. Electr. Comput. Eng.* **2025**, *2025*, 1171569. [CrossRef]
21. Jahandideh, H.; Namvar, M. Use of PSO in parameter estimation of robot dynamics; Part two: Robustness. In Proceedings of the 2012 16th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 12–14 October 2012.
22. Zha, F.; Sheng, W.; Guo, W.; Qiu, S.; Deng, J.; Wang, X. Dynamic parameter identification of a lower extremity exoskeleton using RLS-PSO. *Appl. Sci.* **2019**, *9*, 324. [CrossRef]
23. Lu, H.; Yang, Z.; Zhu, D.; Deng, F.; Guo, S. Dynamics Modeling and Parameter Identification for a Coupled-Drive Dual-Arm Nursing Robot. *Chin. J. Mech. Eng.* **2024**, *37*, 74. [CrossRef]
24. Zhou, M.; Cui, M.; Xu, D.; Zhu, S.; Zhao, Z.; Abusorrah, A. Evolutionary optimization methods for high-dimensional expensive problems: A survey. *IEEE/CAA J. Autom. Sin.* **2024**, *11*, 1092–1105. [CrossRef]
25. Li, C.; Yang, J.; Chang, S. Review on key technologies of space intelligent grasping robot. *J. Braz. Soc. Mech. Sci. Eng.* **2022**, *44*, 64. [CrossRef]

26. Huang, W.; Min, H.; Guo, Y.; Liu, M. A review of dynamic parameters identification for manipulator control. *Cobot* **2022**, *1*, 5. [CrossRef]
27. Erdogmus, P.; Toz, M. Heuristic optimization algorithms in robotics. In *Serial and Parallel Robot Manipulators-Kinematics, Dynamics, Control and Optimization*; Intechopen: London, UK, 2012; pp. 311–338.
28. Fazilat, M.; Zioui, N. Investigating Quantum Artificial Neural Networks for singularity avoidance in robotic manipulators. In Proceedings of the 2024 12th International Conference on Systems and Control (ICSC), Batna, Algeria, 3–5 November 2024.
29. Fazilat, M.; Zioui, N. Quantum Neural Network-based Inverse Kinematics of a Six-jointed Industrial Robotic Arm. *Robot. Auton. Syst.* **2025**, *194*, 105123. [CrossRef]
30. Fazilat, M.; Zioui, N. Quantum-Inspired Sliding-Mode Control to Enhance the Precision and Energy Efficiency of an Articulated Industrial Robotic Arm. *Robotics* **2025**, *14*, 14. [CrossRef]
31. Kuo, C.; Dai, J.S.; Dasgupta, P. Dasgupta, Kinematic design considerations for minimally invasive surgical robots: An overview. *Int. J. Med. Robot. Comput. Assist. Surgery* **2012**, *8*, 127–145. [CrossRef] [PubMed]
32. Corke, P. Image Formation. In *Robotics, Vision and Control: Fundamental Algorithms in MATLAB® Second, Completely Revised, Extended and Updated Edition*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 319–357.
33. Zarychta, D.; Zubrycki, I. Methodology for Teaching the Denavit-Hartenberg Notation. In *International Conference on Robotics in Education (RiE)*; Springer: Berlin/Heidelberg, Germany, 2024.
34. Siciliano, B. Robot Kinematics. In *The Mechanical Systems Design Handbook*; CRC Press Inc.: Boca Raton, FL, USA, 2017; pp. 451–486.
35. Gilbert, H.B.; Rucker, D.C.; Webster, R.J., III. Concentric tube robots: The state of the art and future directions. In *Robotics Research: The 16th International Symposium ISRR*; Springer: Berlin/Heidelberg, Germany, 2016.
36. Sathya, A.S.; Carpentier, J. Constrained articulated body dynamics algorithms. *IEEE Trans. Robot.* **2024**, *41*, 430–449. [CrossRef]
37. Abualigah, L.; Sheikhan, A.; Ikotun, A.M.; Abu Zitar, R.; Alsoud, A.R.; Al-Shourbaji, I.; Hussien, A.G.; Jia, H. Particle swarm optimization algorithm: Review and applications. *Metaheuristic Optim. Algorithms* **2024**, *1*, 1–14. [CrossRef]
38. Lalitha Kameswari, Y. Fundamental of Swarm Optimization. In *Role of Nature-Inspired Algorithms in Real-Life Problems*; Springer: Berlin/Heidelberg, Germany, 2025; pp. 95–117.
39. Gad, A.G. Particle swarm optimization algorithm and its applications: A systematic review. *Arch. Comput. Methods Eng.* **2022**, *29*, 2531–2561. [CrossRef]
40. Shami, T.M.; El-Saleh, A.A.; Alswaitti, M.; Al-Tashi, Q.; Summakieh, M.A.; Mirjalili, S. Particle swarm optimization: A comprehensive survey. *IEEE Access* **2022**, *10*, 10031–10061. [CrossRef]
41. Wang, D.; Tan, D.; Liu, L. Particle swarm optimization algorithm: An overview. *Soft Comput.* **2018**, *22*, 387–408. [CrossRef]
42. Reda, D.; Talaoubrid, A.; Fazilat, M.; Zioui, N.; Tadjine, M. A quantum direct torque control method for permanent magnet synchronous machines. *Comput. Electr. Eng.* **2025**, *122*, 109994. [CrossRef]
43. Feraoun, H.; Fazilat, M.; Dermouche, R.; Bentouba, S.; Tadjine, M.; Zioui, N. Quantum maximum power point tracking (QMPPT) for optimal solar energy extraction. *Syst. Soft Comput.* **2024**, *6*, 200118. [CrossRef]
44. Alvarez-Alvarado, M.S.; Alban-Chacón, F.E.; Lamilla-Rubio, E.A.; Rodríguez-Gallegos, C.D.; Velásquez, W. Three novel quantum-inspired swarm optimization algorithms using different bounded potential fields. *Sci. Rep.* **2021**, *11*, 11655. [CrossRef]
45. Chen, M.; Song, H.; Zhu, D.; Chen, W.; Cai, C. Data Collection in Underwater Acoustic Sensor Networks Using Multiple AUVs with Sequential Greedy-QPSO and Fast Marching Algorithms. *IEEE Sens. J.* **2025**, *25*, 20433–20444. [CrossRef]
46. Fazilat, M.; Zioui, N.; St-Arnaud, J. A novel quantum model of forward kinematics based on quaternion/Pauli gate equivalence: Application to a six-jointed industrial robotic arm. *Results Eng.* **2022**, *14*, 100402. [CrossRef]
47. Yang, C.X.; Zhang, J.; Tong, M.S. A hybrid quantum-behaved particle swarm optimization algorithm for solving inverse scattering problems. *IEEE Trans. Antennas Propag.* **2021**, *69*, 5861–5869. [CrossRef]
48. Zioui, N.; Mahmoudi, A.; Fazilat, M.; Kone, O.; Reda, D.; Tadjine, M. Quantum Space Vector Pulse Width Modulation for Speed Control of Permanent Magnet Synchronous Machines. *E-Prime-Adv. Electr. Eng. Electron. Energy* **2025**, *13*, 101074. [CrossRef]
49. Xu, Z.; Cui, Y.; Li, B. A quantum-inspired particle swarm optimization for sizing optimization of truss structures. *Phys. Conf. Ser.* **2021**, *1865*, 042127. [CrossRef]
50. Sun, J.; Fang, W.; Wu, X.; Palade, V.; Xu, W. Quantum-behaved particle swarm optimization: Analysis of individual particle behavior and parameter selection. *Evol. Comput.* **2012**, *20*, 349–393. [CrossRef] [PubMed]

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