Physics-Informed Learning-based model-free computed torque control for unknown nonlinear rigid dynamic systems

Arman Mardani[1]

***Abstract*-- This paper presents a new model-free control strategy for nonlinear rigid-body systems with unknown or highly complex dynamics. Unlike traditional Computed Torque Control (CTC), which depends on precise analytical models, this approach uses real-time data to estimate system behavior, eliminating the need for prior modeling. The method operates in two key stages. First, it dynamically estimates the Jacobian matrix by applying small perturbations to control inputs, linking task-space motion to joint torques. This Jacobian maps a virtual spring-damper force from the workspace to joint space, ensuring stable trajectory following. Second, a constrained Recursive Least Squares (RLS) algorithm learns a simplified linear representation of the system's nonlinear dynamics, accounting for Coriolis, centrifugal, and gravitational effects. Crucially, this learning process incorporates physical constraints, maintaining energy conservation and other essential rigid-body properties. By combining these techniques, the proposed controller replaces CTC's model-based elements with adaptive, data-driven mechanisms. The result is a real-time compensator for nonlinearities that preserves stability and performance. Tests on a robotic manipulator confirm improved tracking accuracy and robustness over conventional CTC, especially in cases with unmodeled dynamics or parameter uncertainties.**

***Index Terms*-- Numerical Jacobian Estimation, Perturbation-Based Learning, Nonlinear rigid-body systems, dynamic systems approximation.**

I. Introduction

Controlling nonlinear rigid-body systems remains a fundamental challenge in robotics and automation, with critical applications in industrial manipulators [1], surgical robotic systems [2], and space exploration platforms [3]. Since their introduction in the early 1980s [4], model‐based control strategies—particularly computed torque control (CTC)—have been widely regarded as a standard approach for nonlinear robotic systems. These methods rely on precise mathematical representations of system dynamics that incorporate inertial parameters [5], Coriolis and centrifugal effects [6], gravitational forces [7], and friction characteristics [8]. Although these controllers are theoretically well established, their practical performance is often limited by unavoidable modeling inaccuracies and parameter uncertainties [9]. The inherent gap between theoretical models and physical reality has motivated extensive research into control strategies that can operate effectively under imperfect knowledge of system dynamics. Recent advances in adaptive control theory [10] and machine learning techniques [11] have created new opportunities to address these long‐standing challenges. As a result, the robotics community has progressively moved beyond purely model‐based strategies toward hybrid approaches that integrate physical modeling with data‐driven learning mechanisms [12]. This paradigm shift reflects a key observation: while first‐principles modeling provides valuable structural insight into system dynamics [13], practical robotic applications require adaptive mechanisms capable of compensating for uncertainties, disturbances, and time‐varying parameters [14]. A primary limitation of model‐based control approaches is their sensitivity to parameter variations and unmodeled dynamics.

As demonstrated in [15], even small deviations in inertial parameter values can lead to significant tracking errors. Friction modeling represents another major source of uncertainty, since experimental studies have shown that conventional friction models often fail to accurately capture the complex nonlinear behavior observed in real mechanical systems [16]. Furthermore, variations in payload and operating conditions introduce additional uncertainties, which have been widely reported in industrial robotic applications [17]. These limitations are particularly pronounced in scenarios involving high-speed maneuvers, variable payloads, or contact-rich interactions, where dynamic models quickly become outdated or insufficiently accurate. To overcome these limitations, early research efforts focused on the development of robust control techniques. Sliding mode control (SMC), originally proposed in [18], offers strong theoretical guarantees of trajectory tracking in the presence of bounded uncertainties. Nevertheless, practical implementations have revealed several drawbacks, including control chattering [19] and increased actuator effort [20].

Although more recent developments—such as higher‐order sliding mode methods [21] and adaptive neural network–reinforcement learning frameworks [22]—have alleviated some of these issues, fundamental trade‐offs between robustness, smoothness, and implementation complexity continue to persist. These trade-offs often force practitioners to make difficult choices between performance metrics, limiting the widespread adoption of robust methods in applications requiring both precision and energy efficiency.

[1] Corresponding author. Department of Mechanical Engineering, Babol Noshirvani University of Technology, Mazandaran, Iran

, Email: Armanmardani@nit.ac.ir

Contemporary adaptive control methods have evolved significantly beyond their early formulations [23]. Recent surveys [24] highlight three major developments in this field: composite adaptation strategies that exploit multiple error signals [25], neural network–based adaptive control schemes [26], and parameter estimation techniques with provable convergence guarantees [27]. These approaches have demonstrated particular promise in emerging domains such as aerial robotics [28] and soft robotic systems [29]. The common thread among these developments is the recognition that effective adaptation requires not only online parameter adjustment but also careful consideration of the information content in available measurements. In parallel, machine learning techniques have substantially enhanced system identification capabilities. Gaussian process regression, for example, provides probabilistic models with uncertainty quantification that are particularly valuable for safe learning in control applications [30]. Deep neural networks have also shown remarkable function approximation capabilities [31]; however, their deployment in real‑time control systems remain challenging due to computational constraints and stability considerations [32].

More recently, physics‑informed neural networks (PINNs) have been proposed to bridge data‑driven learning with governing physical laws, thereby improving generalization and interpretability [33]. These developments underscore a broader trend toward incorporating structural knowledge into learning frameworks to improve data efficiency and reliability. Impedance control has likewise advanced considerably since the seminal work of Hogan [34]. Modern implementations increasingly incorporate adaptive mechanisms [35] and learning components [36], enabling improved performance in applications such as human–robot collaboration [37] and dynamic object manipulation, including flying object interception [38]. Nevertheless, as noted in [39], most impedance control strategies still rely on accurate kinematic models for effective implementation, which constrains their applicability in uncertain or unstructured environments. Reinforcement learning (RL) has also emerged as a powerful paradigm for robot control [40]. Model‑free RL algorithms [41] have demonstrated impressive performance in simulated environments [42].

However, their practical deployment in physical systems remains limited by challenges related to sample efficiency [43] and safety during exploration [44]. Recent advances in model‑based RL [45] and sim‑to‑real transfer techniques [46, 47] are actively addressing these limitations and improving the feasibility of real‑world applications. Among the most promising recent developments are approaches that integrate model‑based structures with data‑driven adaptation. Examples include physics‑guided machine learning frameworks [48] and recursive least squares (RLS)–based model regression techniques [49]. Building upon these foundations, this work introduces several novel contributions, including a real‑time Jacobian estimation method based on optimized perturbation design [50] and a constrained parameter learning strategy with stability guarantees [51, 52]. Specifically, this paper proposes a comprehensive model‑free computed torque control (CTC) framework that does not require prior knowledge of either the kinematic or dynamic models of the system. The proposed approach incorporates real‑time Jacobian estimation through optimized torque perturbations [53] and a physics‑informed recursive least squares (RLS) parameter estimation scheme with theoretical stability guarantees [54, 55]. The primary objective is to develop a fast, model‑free yet dynamics‑informed adaptive control framework capable of controlling black‑box nonlinear dynamical systems in the task space while the control inputs are applied in the joint space, without requiring any prior knowledge of the system's kinematics or dynamics. The remainder of the paper is organized as follows. Section II reviews classical computed torque control and discusses its fundamental limitations. Section III presents the proposed model‑free control methodology. Section IV provides the stability analysis of the proposed framework. Section V reports the simulation results, and Section VI concludes the paper with discussions and directions for future research.

II. Dynamics and Data-Based Jacobian Approximation

The equations of motion are typically derived in joint space, where the dynamics are expressed in terms of joint angles, velocities, and torques. However, for tasks such as trajectory planning, force control, or human-robot interaction, it is often more convenient to represent the dynamics in task space (or workspace). It describes the motion in Cartesian coordinates (e.g., position and orientation of the end-effector). The configuration of the end-effector in task space are given by the forward kinematics function $\mathbf{x} = f(\mathbf{q})$ where $\mathbf{x} \in R_m$ is the task space variables and $\mathbf{q} \in R_n$ is the joint space coordinate (n degrees of freedom). The end-effector velocity $\dot{\mathbf{X}}$ is related to joint velocities $\dot{\mathbf{q}}$ via the Jacobian matrix $\mathbf{J}(\mathbf{q})$. Differentiating the velocity relation yields the acceleration:

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}} \tag{1}$$

The standard Lagrangian dynamics in joint space and work-space is shown as follows [56]:

$$\begin{aligned} &\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}(\mathrm{q})\dot{\mathbf{q}} + \mathbf{K}(\mathbf{q})\mathbf{q} + \boldsymbol{G}(\mathbf{q}) = \boldsymbol{\tau}, \\ &\mathbf{q} \in R^n \\ &\boldsymbol{\Lambda}(\mathbf{x})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x},\dot{\mathbf{x}}) + \mathbf{J}^{\mathrm{T}}\mathbf{K}\mathbf{J}^{-1}\mathbf{x} + \mathbf{J}^{\mathrm{T}}\mathbf{D}(\mathbf{q})\mathbf{J}^{-1}\dot{\mathbf{q}} + \mathbf{p}(\mathbf{x}) \\ &\qquad = \mathbf{F} \end{aligned} \tag{2}$$

where $\mathbf{M}(\mathbf{q}), \mathbf{C}(\mathbf{q},\dot{\mathbf{q}}), \mathbf{G}(\mathbf{q}), \boldsymbol{\tau}$ respectively represents inertia matrix, Coriolis and centrifugal matrix, gravity vector and joint torque. The equation of motion in work-space equation includes forces in task space F, task-space inertia $\boldsymbol{\Lambda}(\mathbf{x}) = (\mathbf{J}\mathbf{M}(\mathrm{q})^{-1}\mathbf{J}^{\mathrm{T}})^{-1}$, task-space Coriolis/ centrifugal forces $\boldsymbol{\mu}(x,\dot{x}) = \boldsymbol{\Lambda}\dot{\mathbf{J}}\dot{\mathbf{q}} + \boldsymbol{\Lambda}\mathbf{J}\mathbf{M}^{-1}\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}}$ and $\mathbf{p}(x) = \mathbf{J}^{-\mathbf{T}}\mathbf{G}(\mathbf{q})$ task-space gravity effect.

Rigid-body dynamical systems generally follow a common structural form in their equations of motion. This structure provides the foundation for many classical model-based control strategies. However, in practical applications the exact equations of motion may not be available due to incomplete

modeling, parameter uncertainties, or unmodeled nonlinear effects. For systems with linear dynamics, classical adaptive control methods can often estimate key parameters such as the effective inertia (mass) matrix. Although the estimated parameters may not exactly match the true physical parameters of the system, they can still provide satisfactory control performance. In contrast, for nonlinear, time-varying, or structurally uncertain systems, traditional optimal or model-based control approaches may become unreliable due to their strong dependence on accurate system models. Data-driven approaches, such as neural network–based controllers, have been proposed to address these challenges. While these methods offer powerful function approximation capabilities, they typically require extensive training iterations and large amounts of data to capture the true dynamic behavior of the system, which can limit their applicability in real-time control scenarios.

To overcome these limitations, this paper proposes a genuinely model-free control strategy for nonlinear and uncertain dynamical systems. The proposed framework integrates numerical Jacobian estimation, adaptive task-space optimization based on the estimated Jacobian, and a task-space spring–damper control formulation. Unlike traditional model-based controllers that rely on explicit dynamic equations, the proposed approach estimates the manipulator's kinematic and dynamic properties online using finite-difference perturbations combined with least-squares parameter estimation. The resulting numerically estimated task-space dynamic matrices are then incorporated into an approximate computed torque control (CTC) framework, enabling effective control without requiring prior knowledge of the system's kinematic or dynamic models. The proposed kinematics-informed model-free framework relies only on the general structural forms of rigid-body dynamics and kinematics that are common to nonlinear mechanical systems. Consequently, the method does not require analytical Jacobian matrices or explicit dynamic models. The approach is inherently adaptive to changes in system configuration, robust to modeling inaccuracies, and straightforward to implement with minimal computational overhead.

This section presents the mathematical formulation of the proposed method, including the Jacobian estimation procedure, the task-space control law, and the associated stability considerations. The Jacobian is estimated using a finite-difference approach in which small torque perturbations are superimposed on the nominal motor torques. These perturbations generate measurable variations in the system's kinematic response, allowing the relationship between input torques and output motion to be identified and used for online Jacobian estimation. The method tries to find an approximation of $\mathbf{J}(\mathbf{q})$. The Jacobian matrix $\mathbf{J(q)}$ relates joint velocities $\dot{\mathbf{q}}$ to end-effector velocities $\dot{\mathbf{x}}$. Since we don't have kinematics and dynamics matrixes including analytical Jacobian derivation, We estimate the Jacobian numerically by perturbing each joint and observing the resulting displacement of the end effector. For an n-DoF dynamical system, the Jacobian $\mathbf{J}(q)$ can be estimated using a least-squares refinement procedure. To improve robustness, a weighted least-squares approach is employed over a moving window of recent joint configurations and corresponding end-effector positions. The real-time estimator of $\mathbf{J}(q)$ is obtained by solving the following optimization problem.

$$\mathbf{J} = \underset{\mathbf{J}}{\operatorname{argmin}} \sum_{k=1to\,N} \omega_k \, \|\Delta x_k - \mathbf{J}\,\Delta q_k\|^2 \tag{3}$$

where the numerical and sensor-based differences $\Delta x_k, \Delta q_k$ can be monitored when the system is actuated by motor low-amplitude high-frequency modulated torques, and calculate a weighted least squares solution with Tikhonov regularization ($\lambda\mathbf{I}$ term). It solves for the matrix $\mathbf{J}$ in:

$$\mathbf{J} = \underset{\mathbf{J}}{\operatorname{argmin}} \sum_{k=1to\,N} \lambda \, \|\,\mathbf{J}\,\|^2 + \omega_k \, \|\Delta x_k - \mathbf{J}\,\Delta q_k\|^2 \tag{4}$$

The key point is that (4) is a regularized version of the Eq (3) as least□squares problem, which provides a stable and unique solution, especially when the data matrix is ill□conditioned or noisy. The Jacobian matrix is estimated from measured joint and end-effector variations using a least-squares formulation. For a set of N recent measurements, the relationship between the incremental joint motion Δq_k and the corresponding end-effector displacement Δx_k can be approximated as $\Delta x_k \approx J\Delta q_k$. The Jacobian matrix can therefore be estimated by minimizing the weighted prediction error $\mathbf{J} = \underset{\mathbf{J}}{\operatorname{argmin}} \sum_{k=1to\,N} \omega_k \, \|\Delta x_k - \mathbf{J}\,\Delta q_k\|^2$. This formulation corresponds to a weighted least-squares problem where ω_k assigns different importance to each measurement in the sliding data window (which is known as moving window in some other mechatronics field like kernel convolution and so on). However, in practical robotic systems the measurements Δx_k and Δq_k are affected by sensor noise, and the collected samples may be nearly linearly dependent. In such cases the least-squares problem can become ill-conditioned, leading to unstable Jacobian estimates or excessively large parameter values. To improve numerical stability, Tikhonov regularization is introduced, resulting in the modified optimization problem as $\mathbf{J} = \underset{\mathbf{J}}{\operatorname{argmin}} \sum_{k=1to\,N} \lambda \, \|\,\mathbf{J}\,\|^2 + \omega_k \, \|\Delta x_k - \mathbf{J}\,\Delta q_k\|^2$.

III. Workspace Spring-Damper Control

The control objective is to drive the end effector to the desired position $\mathbf{x}_d$ while ensuring smooth and stable convergence. To achieve this, a virtual spring–damper system is defined in the task space:

$$\mathbf{F} = \mathbf{K_p}(\mathbf{x}_d - \mathbf{x}) - \mathbf{K_d}\dot{\mathbf{x}} \tag{5}$$

here $\mathbf{K_p}, \mathbf{K_d}$, x and $\mathbf{K_d}\dot{\mathbf{x}}$ respectively represent stiffness matrix (position gain), damping matrix (velocity gain), current end-effector position and end-effector velocity. To improve

transient response, the gains are adjusted based on tracking error:

$$\mathbf{K_p}(\mathbf{e}) = K_{p0}\left(1 + \alpha \tanh\left(\frac{\|\mathbf{e}\|}{\epsilon}\right)\right),$$
$$\mathbf{K_d}(\mathbf{e}) = K_{p0}\left(1 + \beta \tanh\left(\frac{\|\mathbf{e}\|}{\epsilon}\right)\right) \quad (6)$$

$\mathbf{K_{p0}}$ and $\mathbf{K_{d0}}$ are the nominal proportional and derivative gains, respectively. Also, the symbol □ represents error threshold and α and β are scaling factors for the gain modulation. To eliminate steady-state error, an integral term is added. Final torque of motors can be calculated as $\boldsymbol{\tau} = \mathbf{J}^{\mathbf{T}}\mathbf{F}_{\text{total}}$

$$\mathbf{F}_{total} = \mathbf{K_p}(\mathbf{e})\mathbf{e} + \mathbf{K_i}\int \mathbf{e}\, dt - \mathbf{K_d}(\mathbf{e})\dot{\mathbf{x}} \quad (7)$$

This approach is not a model-based control method, but it exploits a fundamental characteristic of rigid-body dynamics: every such system possesses a Jacobian matrix that maps input torques to output motions, and this Jacobian inherently projects the forces at the end effector into the corresponding motor torques. This property is not limited to special cases; it holds for all rigid-body dynamical systems with defined inputs and outputs. Apparently, this type of virtual force definition yields point-to-point control.

For an undefined system, the explicit relationship between the workspace and joint space is not available. However, using an RLS-based estimation method, it is possible to approximate the Jacobian. This requires a real-time dataset, obtained by applying low-amplitude oscillatory torque perturbations to slightly excite the system and then monitoring the resulting input and output displacements. The next step is to eliminate the contributions of dynamic terms such as Coriolis, centrifugal, gravitational, and inertial effects. In principle, computed torque control (CTC) can be used to compensate for these terms. The main challenge, however, is that CTC is inherently a model-based controller. The contribution of this work is the observation that removing most of the compensation component of the torque can still yield effective control and significantly improve overall performance.

This raises the question of how a model-based strategy can be applied in task space when both the dynamic and kinematic models are unknown. The proposed solution is to approximate the computed-torque term directly from the collected data, enabling a practical data-driven approximation of CTC for systems with undefined dynamics and kinematics. Following equation shows the rigid-body dynamic model $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} + \mathbf{G}(\mathrm{q}) = \boldsymbol{\tau}$. It can be concluded that the torque is a function of $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]$. A real dynamic system can be represented in a simplified form as $\tau_i = f_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ that the subscript *i* denotes the motor *i*.

"This implies that the function $f_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ can be approximated by a hypersurface $S_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \approx f_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$. this hyper surface works more accurate near collected real data $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}]$ which results in real time permanent approximation via regression. This paper presents $\tau_i = S_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ as a hyperplane which is defined as follows [57]:

$$\tau_i = S_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \psi_{i0} + \boldsymbol{\psi}_{i1}\mathbf{q} + \boldsymbol{\psi}_{i2}\dot{\mathbf{q}} + \boldsymbol{\psi}_{i3}\ddot{\mathbf{q}}, \quad \mathbf{q} \in R^n \quad (8)$$

where the vector of weights $\boldsymbol{\psi}_{ij}$ defines the equation of hyperplane and the constant ψ_{i0} represents the constant of the surface. but, we can estimate dynamics-related terms $\boldsymbol{M}_{n\times n}(\mathbf{q}), \mathbf{C}_{n\times n}(\mathbf{q},\dot{\mathbf{q}}), \mathbf{D}_{n\times n}, \mathbf{K}_{n\times n}$ and $\mathbf{G}_{n\times 1}(\mathbf{q})$ respectively as $\widehat{\mathbf{M}}_{n\times n}(\mathbf{q}), \widehat{\mathbf{C}}_{n\times n}(\mathbf{q},\dot{\mathbf{q}}), \widehat{\mathbf{D}}_{n\times n}, \widehat{\mathbf{K}}_{n\times n}$ and $\widehat{\mathbf{G}}_{n\times n}(\mathbf{q})$. Every single element of $\widehat{(.)}(\mathbf{q})$ or $\widehat{(.)}(\mathbf{q},\dot{\mathbf{q}})$ matrix or vector is a hyper-surface approximation $S(\mathbf{q})$ or $S(\mathbf{q},\dot{\mathbf{q}})$ of that element based on numerical data collected from sensors. *n* shows the number of joint-space variables $\mathbf{q} \in R^n$.

$$approximated\ rigid-body\ dynamics:$$
$$\boldsymbol{\tau} = \widehat{\mathbf{M}}_{n\times n}(\mathbf{q})\ddot{\mathbf{q}} + \widehat{\mathbf{C}}_{n\times n}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \widehat{\mathbf{D}}_{n\times n}\mathbf{q} + \widehat{\mathbf{K}}_{n\times n}\mathbf{q} + \widehat{\mathbf{G}}_{n\times n}(\mathbf{q}) \quad (9)$$

The main issue is to find the approximation of $S_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \approx f_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ at first step which is handled via recursive least square (RLS). This probably returns a real-time and updating approximation of $S_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ and consequently, $\widehat{\mathbf{M}}_{n\times n}(\mathbf{q}), \widehat{\mathbf{C}}_{n\times n}(\mathbf{q},\dot{\mathbf{q}}), \widehat{\mathbf{D}}_{n\times n}, \widehat{\mathbf{K}}_{n\times n}$ and $\widehat{\mathbf{G}}_{n\times n}(\mathbf{q})$. Its turn to improve CTC method to numerical case which turns CTC to a model-free method. The dynamics of the manipulator considered in this work can be written in the standard form $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}$. that $\boldsymbol{\tau}$ is the control torque. The proposed controller employs a computed torque compensation structure combined with a task-space PID feedback force, expressed as $\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_d + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} + \mathbf{G}(\mathbf{q}) + \mathbf{J}^{\mathrm{T}}\mathbf{F}_x$ that $\mathbf{F}_x$ is the task space virtual spring damper integrator PID controller written as $\left(\mathbf{K_p}(\mathbf{e})\mathbf{e} + \mathbf{K_i}\int \mathbf{e}\, dt - \mathbf{K_d}(\mathbf{e})\dot{\mathbf{e}}\right)$ where $\mathbf{e} = \mathbf{x_d} - \mathbf{x}$. Considering the kinematic relation $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$ and the error dynamics $\dot{\mathbf{e}} = -\dot{\mathbf{x}}$, the control force behaves as a virtual spring–damper–integrator system in task space. Substituting the control law into the manipulator dynamics leads to $\mathbf{J}^{\mathrm{T}}\mathbf{F}_x = \mathbf{M}(\mathbf{q})\left(-\ddot{\mathbf{q}}_d + \ddot{\mathbf{q}}\right)$ which indicates that the nonlinear dynamics are compensated and the closed-loop behavior is governed by the task-space feedback force. To analyze stability, the Lyapunov candidate function $V = \frac{1}{2}\dot{\mathbf{x}}^{\mathbf{T}}\dot{\mathbf{x}} + \frac{1}{2}\dot{\mathbf{e}}^{\boldsymbol{T}}\mathbf{K_p}\dot{\mathbf{e}} + \frac{1}{2}(\int \boldsymbol{edt})^{\boldsymbol{T}}\mathbf{K_i}(\int \boldsymbol{edt})$ is selected, which is positive definite for positive definite gain matrices $\mathbf{K_d}$, $\mathbf{K_p}$, and $\mathbf{K_i}$. Taking the time derivative of the Lyapunov candidate function gives $\dot{V} = \dot{\mathbf{x}}^{\mathbf{T}}\ddot{\mathbf{x}} + \dot{\mathbf{e}}^{\boldsymbol{T}}\mathbf{K_p}\dot{\mathbf{e}} + (\int \boldsymbol{edt})^{\boldsymbol{T}}\mathbf{K_i}\mathbf{e}$. Considering the kinematic relation $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$ and defining the integral state $\mathbf{z} = \int \mathbf{e}dt$, the derivative becomes $\dot{V} = \dot{\mathbf{x}}^{\mathbf{T}}\ddot{\mathbf{x}} + \mathbf{e}^{\mathbf{T}}\mathbf{K_p}\dot{\mathbf{e}} + \boldsymbol{z}^{\boldsymbol{T}}\mathbf{K_i}\mathbf{e}$. Hence the task-space acceleration can be written as $\ddot{\mathbf{x}} = \mathbf{K_p}\boldsymbol{e} + \mathbf{K_i}\boldsymbol{z} - \mathbf{K_d}\mathbf{x}$. Substituting this expression into $\dot{V}$ yields $\dot{V} = \dot{\mathbf{x}}^{\mathbf{T}}\left(\mathbf{K_p}\boldsymbol{e} + \mathbf{K_i}\boldsymbol{z} - \mathbf{K_d}\mathbf{x}\right) - \mathbf{e}^{\mathbf{T}}\mathbf{K_p}\dot{\mathbf{x}} + \mathbf{z}^{\boldsymbol{T}}\mathbf{K_i}\mathbf{e}$. Rearranging the terms gives $\dot{V} = \dot{\mathbf{x}}^{\mathbf{T}}\mathbf{K_p}\mathbf{e} + \dot{\mathbf{x}}^{\mathbf{T}}\mathbf{K_i}\boldsymbol{z} - \dot{\mathbf{x}}^{\mathbf{T}}\mathbf{K_d}\mathbf{x} - \dot{\mathbf{e}}^{\boldsymbol{T}}\mathbf{K_p}\dot{\mathbf{x}} + \mathbf{z}^{\boldsymbol{T}}\mathbf{K_i}\mathbf{e}$. Since $\mathbf{K_p}$ and $\mathbf{K_i}$ are symmetric positive definite matrices, the cross terms satisfy $\dot{\mathbf{x}}^{\mathbf{T}}\mathbf{K_p}\mathbf{e} = \mathbf{z}^{\boldsymbol{T}}\mathbf{K_i}\mathbf{x}$ and $\dot{\mathbf{x}}^{\mathbf{T}}\mathbf{K_p}\mathbf{e} = \mathbf{e}^{\mathbf{T}}\mathbf{K_p}\dot{\mathbf{x}}$, therefore, the terms $\dot{\mathbf{x}}^{\mathbf{T}}\mathbf{K_p}\mathbf{e} - \mathbf{e}^{\mathbf{T}}\mathbf{K_p}\dot{\mathbf{x}}$ cancel each other. Furthermore, because $\dot{\mathbf{z}} = \mathbf{e}$ and $\dot{\mathbf{e}} = -\dot{\mathbf{x}}$, the remaining mixed terms associated with the integral state are balanced by the stored energy term of the Lyapunov function. After these cancellations, the derivative of the Lyapunov function reduces to $\dot{V} = -\dot{\mathbf{x}}^{\mathbf{T}}\mathbf{K_p}\dot{\mathbf{x}}$. So, for positive

definite $\mathbf{K_p}$, the quadratic form $\dot{\mathbf{x}}^{\mathrm{T}}\mathbf{K_p}\mathbf{x}$ is strictly positive and $-\dot{\mathbf{x}}^{\mathrm{T}}\mathbf{K_p}\dot{\mathbf{x}}$ is negative. Consequently, the Lyapunov function is non-increasing and bounded from below, ensuring that all closed-loop signals remain bounded. Following equation is the CTC control law [58]:

$$\boldsymbol{\tau} = (\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}}_{des} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{D}\dot{\mathbf{q}} + \mathbf{K}\mathbf{q} + \mathbf{G}(\mathbf{q})) + \left(J^{\mathrm{T}}(\mathbf{K_p}(\mathbf{e})\mathbf{e} + \mathbf{K_i}\int \mathbf{e}\ dt - \mathbf{K_d}(\mathbf{e})\dot{\mathbf{e}})\right) \quad (10)$$

From approximation:

$$\hat{\boldsymbol{\tau}} = \left(\widehat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_{des} + \hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \widehat{\mathbf{D}}\dot{\mathbf{q}} + \widehat{\mathbf{K}}\mathbf{q} + \widehat{\mathbf{G}}(\mathbf{q})\right) + \left(\hat{J}^{\mathrm{T}}(\mathbf{K_p}(\mathbf{e})\mathbf{e} + \mathbf{K_i}\int \mathbf{e}\ dt - \mathbf{K_d}(\mathbf{e})\dot{\mathbf{e}})\right) \quad (11)$$

The note is that $\hat{\boldsymbol{\tau}}$ is not permanently an accurate approximation of $\boldsymbol{\tau}$. its just for the hyper points close the mean of the points $[\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}}]$ that are used for RLS weighted method to approximate dynamic terms in a limited time interval. It means the approximation has to be continued in all times of dynamic system motion. The estimated terms of the equation $\hat{\boldsymbol{\tau}} = \left(\widehat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_{des} + \hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \widehat{\mathbf{D}}\dot{\mathbf{q}} + \widehat{\mathbf{K}}\mathbf{q} + \widehat{\mathbf{G}}(\mathbf{q})\right)$ are all required numerical approximations to turn CTC control to a model-free method that can be applied for the systems that featured by following conditions. The first, rigidity of the mechanism or dynamic system, the second, the systems with enough time to be estimated.

IV. Model-free data-driven dynamics-informed CTC

Although the proposed dynamics compensation strategy effectively mitigates the nonlinear effects present in the system, it does not explicitly enforce all intrinsic structural properties of rigid-body dynamics. The parameter estimation mechanism relies on a standard recursive least-squares (RLS) algorithm, which is widely used in adaptive control frameworks.

The principal contribution of the present approach lies not in the estimator itself, but in the manner in which the regressor approximation is constructed and organized. This formulation enables the controller to capture the dominant dynamic behavior without requiring an explicit analytical model of the manipulator. The overall control procedure implemented in this study is summarized in the flowchart presented in Fig. 1.

It's explicitly designed to respect the underlying physics described by the basic dynamic equations $\mathbf{M}(\mathrm{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathrm{q},\dot{\mathrm{q}})\dot{\mathbf{q}} + \mathbf{D}(\mathrm{q})\dot{\mathbf{q}} + \mathbf{K}(\mathrm{q})\mathbf{q} + \boldsymbol{G}(\mathbf{q}) = \boldsymbol{\tau}$**.** The form of estimated equation of motion is expressed as. $\hat{\boldsymbol{\tau}} = \left(\widehat{\mathbf{M}}(\mathbf{q})\ddot{\mathbf{q}}_{des} + \hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \widehat{\mathbf{D}}\dot{\mathbf{q}} + \widehat{\mathbf{K}}\mathbf{q} + \widehat{\mathbf{G}}(\mathbf{q})\right)$.

A mechanical engineer is generally familiar with the inherent constraints and structural characteristics associated with this general form of the dynamic equation. These constraints arise from the physical properties of mechanical systems and the fundamental structure of rigid-body dynamics. For clarity, the principal constraints are summarized in the following set of expressions, as reported in [59]:

$$\begin{cases} \mathrm{ST}_1: \ \widehat{\mathbf{M}}, \widehat{\mathbf{D}}, \widehat{\mathbf{K}} = \widehat{\mathbf{M}}^{\mathbf{T}}, \widehat{\mathbf{D}}^{\mathrm{T}}, \widehat{\mathbf{K}}^{\mathrm{T}} \\ \mathrm{ST}_2: \left[\dot{\widehat{\boldsymbol{M}}} - 2\widehat{\boldsymbol{C}}\right] = -\left[\dot{\widehat{\boldsymbol{M}}} - 2\widehat{\boldsymbol{C}}\right]^{T} \end{cases} \quad (12)$$

In addition, the term $\mathbf{C}(\mathrm{q},\dot{\mathrm{q}})$ is generally resulted from $\dot{\mathbf{M}}(\mathrm{q})$ as the following equation [59]:

$$\hat{\mathbf{C}}_{ij} = \frac{1}{2}\sum_{k=1}^{n}\left(\frac{\partial \widehat{\mathbf{M}}_{ij}}{\partial q_k} + \frac{\partial \widehat{\mathbf{M}}_{ik}}{\partial q_j} - \frac{\partial \widehat{\mathbf{M}}_{jk}}{\partial q_i}\right)\dot{\mathbf{q}}_k \quad (13)$$

The use of a dynamics-informed regressor obtained through a modified weighted recursive least-squares (RLS) formulation makes the resulting approximation of the system dynamics more physically interpretable and consistent with the fundamental characteristics of the general equations of motion. By embedding these structural properties into the regressor, the approximated dynamics retain key mechanical features of the underlying system rather than behaving as a purely numerical fit. Building upon this idea, a data-driven dynamics approximation based on a constrained, dynamics-informed recursive least-squares (RLS) scheme is employed to transform the classical computed torque control (CTC) framework into an adaptive, model-free controller. This formulation allows the controller to estimate the required dynamic terms directly from measured data while preserving the essential structure of the mechanical dynamics.

Consider $\mathbf{M}(\mathrm{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathrm{q},\dot{\mathrm{q}})\dot{\mathbf{q}} + \mathbf{D}(\mathrm{q})\dot{\mathbf{q}} + \mathbf{K}(\mathrm{q})\mathbf{q} + \boldsymbol{G}(\mathrm{q}) = \boldsymbol{\tau}$ as the general form of equation of the system where $\mathbf{q} \in R^n$. The overall form of RLS-related matrix is $\hat{\boldsymbol{\tau}} = \mathbf{Y}\boldsymbol{\Theta}$ where $\mathbf{Y} \in \mathrm{R}^{n\times p}$ is the regression matrix and $\boldsymbol{\Theta} \in \mathrm{R}^{p}$ is the vector of unknown parameters and p the number of parameters. We aim to estimate the parameters $\boldsymbol{\Theta}$ of an unknown *n*-DOF rigid-body system from data set filled with rows $[\mathbf{q},\dot{\mathbf{q}},\ddot{\mathbf{q}},\boldsymbol{\tau}]$. the number of unknown parameters has to be calculated Based on the symmetry property of the inertia $\widehat{\mathbf{M}}$ and the commonly imposed symmetric positive-definite structure of the stiffness $\widehat{\mathbf{K}}$ and damping $\widehat{\mathbf{D}}$ matrices. Respectively, the number of total unknown parameters is $n(n+1)(n+1)/2$ for $\widehat{\mathbf{M}}$ matrix because Each $\widehat{\mathbf{M}}_{ij}(\boldsymbol{q})$ is a linear combination of basis functions $\boldsymbol{\phi}_k(\mathbf{q}) = [1, q_1, \dots, q_n]$. Indeed, each $\widehat{\mathbf{M}}_{ij}(\boldsymbol{q})$ has n+1 unknown parameter because the basic function $\boldsymbol{\phi}_k(\mathbf{q}) = [1, q\_1, \dots, q_n]_{1\times(1+n)}$ has n+1 element which is combined to unknown parameter as $\widehat{\mathbf{M}}_{ij} = [\boldsymbol{\theta}]_{ij}^{\mathbf{T}}\boldsymbol{\Phi}(q)$ wherein $[\boldsymbol{\theta}]_{ij}^{\mathbf{T}}$ is the vector of unknown parameters to determine the regression

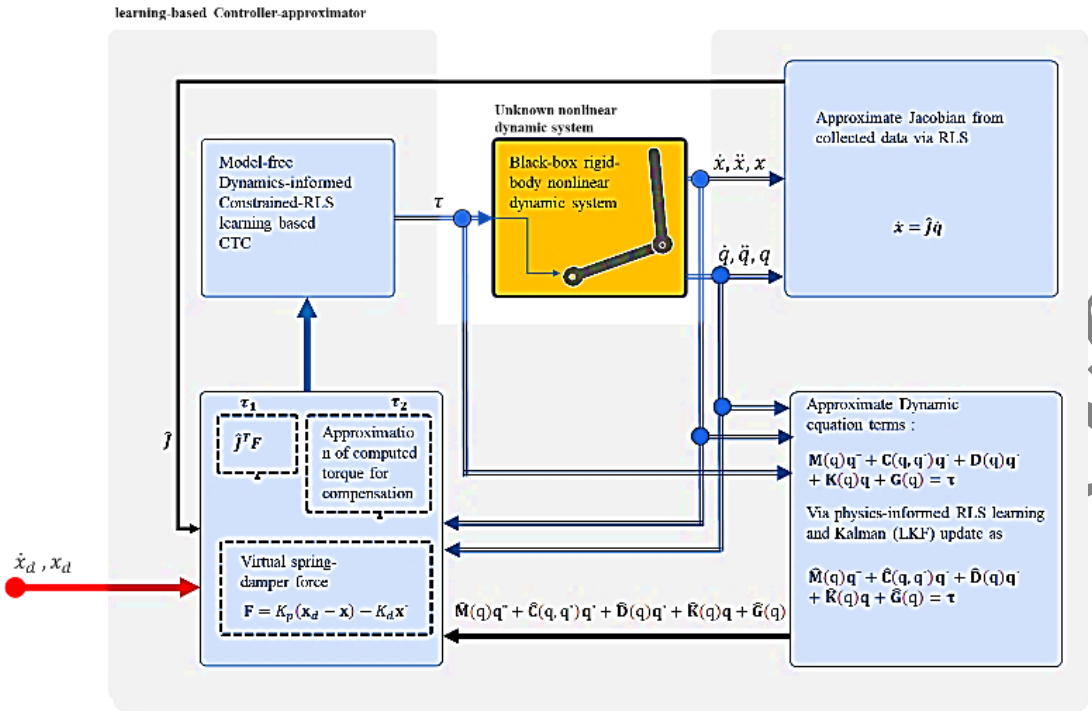

Fig. 1. CTC proposed control

function for the element M_{ij}. So, the $[\boldsymbol{\theta}]_{ij}$ is a 1-by-n+1 vector and $\boldsymbol{\theta}_{ij}^{k}$ is the k_{th} element of the vector $[\boldsymbol{\theta}]_{ij}$. The stiffness and damping matrix is assumed as constant matrix in this case. Consequently, the number of their unknown parameters is $n(n+1)/2$. the gravity term has $n(n+1)$because each element of this vector is unknown regression function with n+1 unknown parameters if the basic function is assumed as $[1, q_1, \dots, q_n]$. The acceleration-related terms can be defined as:

$$\left(\hat{\mathbf{M}}\ddot{q}\right)_i = \sum_{j=1}^{n} \hat{\mathbf{M}}_{ij}\ddot{\mathbf{q}}_j = \sum_{j=1}^{n}\left(\sum_{k=1}^{n+1} \boldsymbol{\theta}_{ij}^{k}\, \phi(\mathbf{q})_k\right)\ddot{\mathbf{q}}_j \tag{14}$$

The term $\left(\hat{\mathbf{M}}\ddot{q}\right)_i$ shows the result of matrix multiplication of $\hat{\mathbf{M}}_{ij}\ddot{q}_j$ to find a line of n-DoF equation of dynamic motion. Also, $\phi(\mathbf{q})_k$ shows the element k_{th} of the vector $\phi(\mathbf{q})$. The regressor matrix rows can be defined as follows:

$$\begin{aligned}
&(\mathbf{Y}_M)_{row=i} = [\, part_1 \,|part_2\,|\, part_3\,] \\
&\mathbf{O}_u = \boldsymbol{O}_{1\times(n+1)} \\
&part_1 = \vartheta(\mathbf{O}_u, i-1) \\
&part_2 = [\ddot{q}_1.[1, q_1, \dots, q_n]\ , \ \dots \ , \\
&\qquad \ddot{q}_n.[1, q_1, \dots, q_n]\ ]_{n\times(1+n)} \\
&\qquad = [\mathbf{Y_{r1}}|\dots|\mathbf{Y}_{rn}] \\
&part_3 = \vartheta(\mathbf{O}_u, n+1-i) \\
&\mathbf{Y}_{M_{row=1}} \\
&= [\mathbf{Y_{r1}}\ \ \mathbf{Y}_{r2}\ \ \dots\ \ \mathbf{Y}_{rn}\ \ \mathbf{O}_u\ \ \mathbf{O}_u\ \ \dots.\ \ \mathbf{O}_u\ \ \mathbf{O}_u\,] \\
&\mathbf{Y}_{M_{row=2}} \\
&= [\mathbf{O}_u\ \ \mathbf{Y_{r1}}\ \ \mathbf{Y}_{r2}\ \ \dots\ \ \mathbf{Y}_{rn}\ \ \mathbf{O}_u\ \ \dots.\ \ \mathbf{O}_u\ \ \mathbf{O}_u] \\
&\mathbf{Y}_{M_{row=n}} \\
&= [\mathbf{O}_u\ \ \mathbf{O}_u\ \ \dots\ \ \mathbf{O}_u\ \ \mathbf{O}_u\ \ \mathbf{Y_{r1}}\ \ \mathbf{Y}_{r2}\ \ \dots\ \ \mathbf{Y}_{rn}]
\end{aligned} \tag{15}$$

$(\mathbf{Y}_M)_{row=i}$ shows the i_{th} row of the regressor matrix which is the essential property of RLS method. Each row of matrix $(\mathbf{Y}_M)_{row=i}$ is defined as $\mathbf{Y}_{M_{row=1\ to\ n}}$. The details of what exactly forms this matrix will be expressed in next equations ((18)), where $\vartheta(\mathbf{a}, b)$ is the function to pattern a matrix $\mathbf{a}$ horizontally b times. In a similar approach, the regression matrix can be defined for stiffness $\mathbf{Y}_K$, damping $\mathbf{Y}_D$ and gravity $\mathbf{Y}_G$. Following formula shows how to combine all together to find the final regression matrix [60]:

$$\begin{aligned}
&\hat{\tau} = \mathbf{Y}\boldsymbol{\Theta} \\
&\mathbf{Y} = [\mathbf{Y}_M|\mathbf{Y}_C|\,\mathbf{Y}_D|\mathbf{Y}_K|\mathbf{Y}_G]\,, \quad \boldsymbol{\Theta} = \left[\boldsymbol{\Theta}_{\mathbf{m}}^{T}|\boldsymbol{\Theta}_{\mathbf{C}}^{T}|\boldsymbol{\Theta}_{\mathbf{D}}^{T}|\boldsymbol{\Theta}_{\mathbf{K}}^{T}|\boldsymbol{\Theta}_{\mathbf{g}}^{T}\right]^{T}
\end{aligned} \tag{16}$$

Next equation Eq (17) clearly defines the parameter of this formulation. Selecting just the upper elements of $\hat{\mathbf{K}}$, $\hat{\mathbf{M}}$, and $\hat{\mathbf{D}}$ matrix is due to the symmetry of this terms which reduces the number of required elements of coefficients in RLS. $[\mathbf{Y}_M|\mathbf{Y}_C|\,\mathbf{Y}_D|\mathbf{Y}_K|\mathbf{Y}_G]$ are the augmented regression matrix. For example, a complete solution for a 2D 2-link manipulator which our main case study can be summarized as follows if the linier basic function and the variables respectively are $\phi_k(q) = [1, q_1, q_2]$ and $[q_1, q_2]$:

$$\begin{aligned}
&\hat{\mathbf{M}}(\mathrm{q})\ddot{\mathbf{q}} + \hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \hat{\mathbf{D}}(\mathbf{q})\dot{\mathbf{q}} + \hat{\boldsymbol{K}}(\mathbf{q})\mathbf{q} + \cdots \\
&\qquad \hat{\mathbf{G}}(\mathbf{q}) = \boldsymbol{\tau} \rightarrow \hat{\boldsymbol{\tau}} \\
&\hat{\mathbf{M}} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} = m_{12} & m_{22} \end{bmatrix} \\
&= \begin{bmatrix} \theta_{11}^{1} + \theta_{11}^{2} q_1 + \theta_{11}^{3} q_2 & \theta_{12}^{1} + \theta_{12}^{2} q_1 + \theta_{12}^{3} q_2 \\ symmetric & \theta_{22}^{1} + \theta_{22}^{2} q_1 + \theta_{22}^{3} q_2 \end{bmatrix}
\end{aligned} \tag{17}$$

$$\hat{\mathbf{D}} = \begin{bmatrix} d_{11} & d_{12} \\ d_{21} = d_{12} & d_{22} \end{bmatrix} = constant\ and\ unknown$$

$$\hat{\mathbf{K}} = \begin{bmatrix} k_{11} & k_{12} \\ d_{21} = d_{12} & k_{22} \end{bmatrix}$$

$$\widehat{\mathbf{G}} = [g_1 \quad g_2]^T$$
$$= \left[\theta_{g1}^1 + \theta_{g1}^2 q_1 + \theta_{g1}^3 q_2 \quad \theta_{g2}^1 + \theta_{g2}^2 q_1 + \theta_{g2}^3 q_2\right]^T$$
$$\boldsymbol{\theta}_{\mathrm{M}} = \left[\theta_{11}^1\ \theta_{11}^2\ \theta_{11}^3 | \theta_{12}^1\ \theta_{12}^2\ \theta_{12}^3 | \theta_{22}^1\ \theta_{22}^2\ \theta_{22}^3\right]$$
$$\boldsymbol{\theta_D}, \boldsymbol{\theta_k}, \boldsymbol{\theta_g} = [d_{11}, d_{12}, d_{22}], [k_{11}, k_{12}, k_{22}], \ldots$$
$$\left[\theta_{g1}^1\ \theta_{g1}^2\ \theta_{g1}^3 \mid \theta_{g2}^1\ \theta_{g2}^2\ \theta_{g2}^3\right]$$
$$\mathbf{Y}_{\mathrm{M}} = [Y_{m1}\ \ Y_{m2}\ ] \in R^{2\times 9}$$
$$\mathbf{Y}_{m1} = \begin{bmatrix} \ddot{q}_1 & \ddot{q}_1 q_1 & \ddot{q}_1 q_2 \\ 0 & 0 & 0 \end{bmatrix} \in R^{2\times 9}$$
$$\mathbf{Y}_{m2} = \begin{bmatrix} \ddot{q}_2 & \ddot{q}_2 q_1 & \ddot{q}_2 q_2 \\ \ddot{q}_1 & \ddot{q}_1 q_1 & \ddot{q}_1 q_2 \end{bmatrix} \in R^{2\times 9}$$
$$\mathbf{Y}_D, \mathbf{Y}_k, \qquad \mathbf{Y}_g = \begin{bmatrix} \dot{q}_1 & \dot{q}_2 & 0 \\ 0 & \dot{q}_1 & \dot{q}_2 \end{bmatrix}, \ldots \begin{bmatrix} q_1 & q_2 & 0 \\ 0 & q_1 & q_2 \end{bmatrix}, \ldots$$
$$\begin{bmatrix} 1 & q_1 & q_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & q_1 & q_2 \end{bmatrix}$$

As mentioned previously, the Coriolis/centrifugal term $\mathbf{C}(\mathrm{q}, \dot{\mathrm{q}})\dot{\mathbf{q}}$ is not explicitly estimated in the standard regressor formulation. it is derived from $\widehat{\mathbf{M}}$ **(q).** since the basic function is supposed to be linear, the term $\hat{\mathbf{C}}(\mathrm{q}, \dot{\mathrm{q}})\dot{\mathbf{q}}$can be calculated as follows:

$$\widehat{\mathbf{M}} = \begin{bmatrix} \theta_{11}^1 + \theta_{11}^2 q_1 + \theta_{11}^3 q_2 & \theta_{12}^1 + \theta_{12}^2 q_1 + \theta_{12}^3 q_2 \\ symmetric & \theta_{22}^1 + \theta_{22}^2 q_1 + \theta_{22}^3 q_2 \end{bmatrix} \tag{18}$$
$$\left(\hat{\mathbf{C}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}\right)_i = \frac{1}{2}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\frac{\partial \widehat{\mathbf{M}}_{ij}}{\partial q_k} + \frac{\partial \widehat{\mathbf{M}}_{ik}}{\partial q_j} - \frac{\partial \widehat{\mathbf{M}}_{jk}}{\partial q_i}\right)\dot{q}_k \dot{q}_j$$

Substituting matrix M and applying derivation based on linear basic function for 2DoF system, the term $(\mathbf{C}(q, \dot{q})\dot{\mathbf{q}})$ can be finally written as:

$$\mathbf{Y}_C = [\mathbf{Y}_{C1} | \mathbf{Y}_{C2}], \boldsymbol{\theta_C} = \boldsymbol{\theta}_{\mathrm{M}}$$
$$\mathbf{Y}_{C1} = \begin{bmatrix} 0 & 0 & \dot{q}_1\dot{q}_2 - \dot{q}_2^2 \\ 0 & -\frac{1}{2}\dot{q}_1\dot{q}_2 & 0 \end{bmatrix} \tag{19}$$
$$\mathbf{Y}_{C2} = \begin{bmatrix} 0 & -\frac{1}{2}\dot{q}_1\dot{q}_2 & 0 \\ 0 & \dot{q}_1\dot{q}_2 - \dot{q}_1^2 & 0 \end{bmatrix}$$

$\boldsymbol{\theta_C} = \boldsymbol{\theta}_{\mathrm{M}}$ because $\hat{\mathbf{C}}$ depends on $\widehat{\mathbf{M}}$, not an independent term. This is one of the most important dynamics-informed constraints for RLS. The not is that when we approximate the system dynamics based on its DoF, all the systems return similar $\mathbf{Y}_{(.)}$. which means it doesn't depend on the exact model. Final torque approximation is:

$$\hat{\boldsymbol{\tau}} = [\mathbf{Y}_M | \mathbf{Y}_C |\ \mathbf{Y}_D | \mathbf{Y}_K | \mathbf{Y}_G]\left[\boldsymbol{\Theta}_{\mathbf{m}}^T | \boldsymbol{\Theta}_{\mathbf{C}}^T | \boldsymbol{\Theta}_{\boldsymbol{D}}^T | \boldsymbol{\Theta}_{\boldsymbol{K}}^T | \boldsymbol{\Theta}_{\boldsymbol{g}}^T\right]^T \tag{20}$$

This approximation is linear but strictly guaranties the dynamics constraints cush as symmetry of $\widehat{\mathbf{M}}$, $\widehat{\mathbf{D}}$ and $\widehat{\mathbf{K}}$ and $\left[\dot{\widehat{\mathbf{M}}} - 2\hat{\mathbf{C}}\right] = -\left[\dot{\widehat{\mathbf{M}}} - 2\hat{\mathbf{C}}\right]^T$. In addition, the $\hat{\mathbf{C}}$ is not independent and resulting from $\widehat{\mathbf{M}}$ based on (18).

If the system is simulated or works in the real world, the data set $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\tau}]$ is collected each time. Following pseudocode illustrates RLS real time update. The Kalman filter can be adapted to perform Recursive Least Squares (RLS) estimation, providing an elegant solution for online parameter estimation in dynamic systems. The standard RLS problem can be framed as a special case of the Kalman filter where the parameters to be estimated are treated as constant states (no process noise) and the measurements are linear combinations of these parameters [61].

1- Initiate parameter $\widehat{\boldsymbol{\Theta}}_0 = \boldsymbol{O}$ if it's not the first-time step, use the last values resulting from previous time step.
2- Set covariance matrix P_0=αI, where α is large (e.g., 1e6) to allow rapid initial adaptation
3- Set forgetting factor λ∈(0,1] (0.95≤λ≤1)
4- Measure $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\tau}]$ via simulation or detect from real sensors
5- Compute error $\boldsymbol{e_k} = \boldsymbol{\tau_k} - \boldsymbol{Y_k}\widehat{\boldsymbol{\Theta}}_{k-1}$
6- Compute Kalman gain $\boldsymbol{K_k} = \boldsymbol{P_{k-1}}\boldsymbol{Y_k^T}(\lambda + \boldsymbol{Y_k}\ \boldsymbol{P_{k-1}}\boldsymbol{Y_k^T})^{-1}$
7- Update parameter and covariance $\widehat{\boldsymbol{\Theta}}_k = \widehat{\boldsymbol{\Theta}}_{k-1}\mathbf{K_k}\mathbf{e_k}$, $\mathbf{P}_k = \frac{1}{\boldsymbol{\lambda}}(\mathbf{I} - \mathbf{K_k}\mathbf{Y_k})\mathbf{P}_{k-1}$

(21)

This pseudocode shows the Kalman filter implementation to solve RLS n-dimensional matrix-based problems. The measurement vector $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\tau}]$ comes from sensor in ideal problem that we know the joint space. In our black box system, we define it as the input motor dynamic property and assume that we them via sensor feedback. So, we can compute error of expected torque and the real one as $\boldsymbol{e_k} = \boldsymbol{\tau_k} - \boldsymbol{Y_k}\widehat{\boldsymbol{\Theta}}_{k-1}$, where and $\boldsymbol{Y_k}$ are $\widehat{\boldsymbol{\Theta}}_{k-1}$ respectively the regressor and the parameter to be determined. Indeed , the term $[\mathbf{Y}_M | \mathbf{Y}_C |\ \mathbf{Y}_D | \mathbf{Y}_K | \mathbf{Y}_G]\left[\boldsymbol{\Theta}_{\mathbf{m}}^T | \boldsymbol{\Theta}_{\mathbf{C}}^T | \boldsymbol{\Theta}_{\boldsymbol{D}}^T | \boldsymbol{\Theta}_{\boldsymbol{K}}^T | \boldsymbol{\Theta}_{\boldsymbol{g}}^T\right]^T$ that is summarized as $\boldsymbol{Y_k}$ $\widehat{\boldsymbol{\Theta}}_{k-1}$ is the last update of regressor-parameter (as RLS needs) . in line 6 of pseudocode, the Kalman gain $\boldsymbol{K_k} = \boldsymbol{P_{k-1}}\boldsymbol{Y_k^T}(\lambda + \boldsymbol{Y_k}\ \boldsymbol{P_{k-1}}\boldsymbol{Y_k^T})^{-1}$is computed. The propagation matrix is a time-varying matrix that estimates the error of Kalman filter and Kalman estimator [61]. It's the fundamental of Kalman method to update parameters of regression. It is also called covariance matrix of the error. At the beginning of the simulation, it is set on a diagonal initial matrix with Hight values which implies that the variance of expectation error is high. Finally, the regression parameters $\widehat{\boldsymbol{\Theta}}_{k-1}$ are updated to $\widehat{\boldsymbol{\Theta}}_k$. Obviously, it's a recursive process during time. Kalman filter is essentially an optimal estimator (the optimization of the formulas that yields these final form can be followed in [61]).

V. Simulation and Results

This simulation implements a model-free adaptive controller for a 2-link robotic manipulator with black-box dynamics. The controller uses Online Jacobian estimation via weighted least squares, Workspace virtual forces with adaptive

PID gains and Joint torque control with smooth saturation. Following table shows the properties of simulation.

Table I
Dynamics and kinematics of the case study

Parameter	Value	Units	Description
L1, L2	0.5	m	Link lengths
m1, m2	1.0	kg	Link masses
C	3.0	N·m·s/rad	Joint damping
g	9.81	m/s²	Gravity
dt	0.001	s	Time step
Kp	400	N·m/rad	Proportional gain
Kd	20	N·m·s/rad	Derivative gain
Ki	20	N·m/(rad·s)	Integral gain
Max Torque	18.0	N·m	Saturation limit

The advanced control method employs recursive least squares (RLS) with exponential weighting for Jacobian estimation, using an 8-sample window and linearly weighted data (0.5 to 1.5). Virtual force computation combines adaptive PID in task space, with gain scheduling via a tanh(error) function and anti-windup integral control. Torque mapping applies a smooth Jacobian transpose (J□F) transformation, incorporating tanh-based saturation and small noise (±0.001 Nm) for realism. This structured approach ensures precise dynamic adaptation while maintaining stability. The method balances responsiveness and smoothness, making it suitable for high-performance robotic control under varying conditions. Figure1 illustrates the result of controlling black-box via Jacobean approximation, work-space virtual approaching force and PID controller in joint space. The result of the proposed approximation shown in Fig.2 demonstrates the efficiency of enhanced model while compared with PID controller.

The proposed controller architecture comprises three main components implemented in a cascade structure. The Jacobian Estimation module employs a recursive least squares (RLS) algorithm with exponential weighting, utilizing a window size of 8 and weighting recent data within a range of 0.5 to 1.5 to maintain adaptability while filtering noise. The Virtual Force component implements an adaptive PID controller in the workspace, featuring gain scheduling through a tanh(error) function for smooth gain transitions and an anti-windup mechanism on the integral term to prevent integrator saturation. Finally, the Torque Mapping module transforms the virtual force from the workspace to joint torques via the J□F transformation, incorporating smooth saturation using a tanh function to limit output and adding controlled noise of ±0.001 Nm for robustness testing. This modular architecture enables real-time adaptive control without requiring prior kinematic or dynamic models, with each component addressing a specific aspect of the model-free control strategy while maintaining computational efficiency and numerical stability.

Also , The constrained recursive least squares (RLS) estimator used in this work is implemented in the form of a Kalman□filter–based parameter estimator, where the unknown dynamic parameters are treated as constant states with zero process noise. At the beginning of the identification process, the parameter estimate vector is initialized as $\widehat{\boldsymbol{\Theta}}_0 = \boldsymbol{0}$, representing the absence of prior knowledge about the system parameters. The covariance matrix is initialized as $\mathbf{P}_0 = \alpha \mathbf{I}$, where I is the identity matrix and α is selected as a large scalar ($\alpha = 10^6$) in order to represent high initial uncertainty and allow rapid adaptation during the early phase of estimation. A forgetting factor λ is introduced to provide robustness to modeling errors and slow parameter variations; in practice λ is chosen close to unity with $0.95 \leq \lambda \leq 1$. At each sampling instant k, the measurement vector consisting of the joint variables and torque signals $[\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}, \boldsymbol{\tau}]$ is obtained either from simulation or from physical sensors. Using these measurements, the regressor matrix $\mathbf{Y}_k$ is constructed and the predicted torque is computed from the previous parameter estimate as $\boldsymbol{Y}_k \widehat{\boldsymbol{\Theta}}_{k-1}$. The prediction error is then calculated as $\mathbf{e}_k = \boldsymbol{\tau}_k - \boldsymbol{Y}_k \widehat{\boldsymbol{\Theta}}_{k-1}$. Based on this error, the Kalman gain is determined as $\boldsymbol{P}_{k-1}\boldsymbol{Y}_k^T\left(\lambda + \boldsymbol{Y}_k \boldsymbol{P}_{k-1} \boldsymbol{Y}_k^T\right)^{-1}$, which regulates the magnitude of the parameter correction. The parameter vector is subsequently updated according to $\widehat{\boldsymbol{\Theta}}_k = \widehat{\boldsymbol{\Theta}}_{k-1} \mathbf{K}_k \mathbf{e}_k$, while the covariance matrix is recursively propagated as $\mathbf{P}_k = \frac{\mathbf{1}}{\boldsymbol{\lambda}}(\mathbf{I} - \mathbf{K}_\mathbf{k}\mathbf{Y}_\mathbf{k})\mathbf{P}_{k-1}$. In the proposed framework, the regressor–parameter product $\boldsymbol{Y}_k \widehat{\boldsymbol{\Theta}}_{k-1}$ corresponds to the grouped dynamic model composed of inertial, Coriolis, damping, stiffness, and gravity components, i.e., $[\mathbf{Y}_M|\mathbf{Y}_C|\,\mathbf{Y}_D|\mathbf{Y}_K|\mathbf{Y}_G]$. The covariance matrix P□, also known as the propagation matrix, represents the evolving uncertainty of the parameter estimates and gradually decreases as the estimator converges. Through this recursive update mechanism, each new measurement incrementally refines the parameter vector without requiring storage of past data, enabling efficient online identification of the unknown parameters of the two□degree□of□freedom linkage.

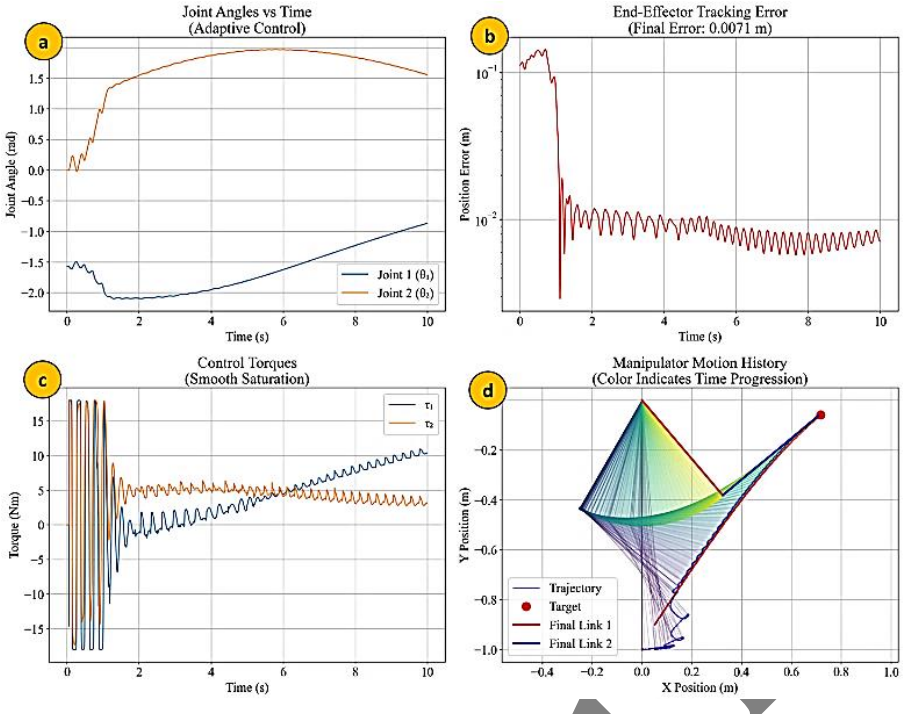

Fig. 2. PID controller + Jacobean approximation. (a) joint angles of 2D 2link manipulator, (b) position distance error, (c) torque of motors, (d) X-Y illustration

Figure 2 shows the joint angles have undesired fluctuation at the begging of motion (a) which is due to unstable Jacobian approximation during the first time steps (d). the error results in (b) obviously imply the chattering trend of movement. Its resulting from pure PID implementation in n-DoF unknown nonlinear system. This simulation runs subject to PID coefficients = 200,60, 30. Other simulation illustrated in Fig. 3 shows the better results for P,I,D coefficients 400, 20, 20. The main question is that what is the best setup to improve controller? This yields us to use better controller specially the one compensates dynamic terms. The first barrier is that computed torque method is not a model-free controller. This contribution tries to develop a model-free CTC.

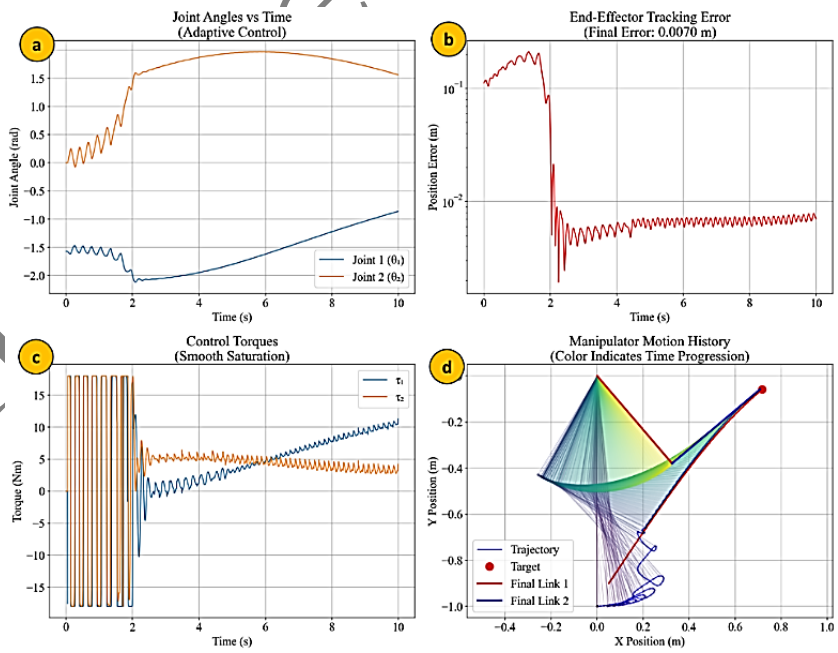

Fig. 3. PID controller + Jacobean approximation based on P,I,D= 400,20,20. (a) joint angles of 2D 2link manipulator, (b) position distance error, (c) torque of motors, (d) X-Y illustration

A. Real-time Jacobian and classic adaptive control

This study presents a comparative analysis of two adaptive control strategies for a two-link robotic manipulator with identical link lengths and masses. The advanced approach combines adaptive PID control with online Jacobian estimation, utilizing a moving window least-squares technique with an 8-sample window size to continuously update the manipulator's Jacobian matrix. While demonstrating superior handling of system nonlinearities and inter-joint coupling effects, this enhanced performance increases computational load due to the real-time Jacobian updates. In contrast, the basic control method implements a conventional adaptive PID scheme with simplified assumptions, including a fixed identity matrix approximation of the Jacobian and a single adaptive gain parameter that scales with tracking error. Though easier to implement, this approach shows noticeable limitations in convergence speed, steady-state accuracy, and its ability to compensate for dynamic coupling between joints, often resulting in overshoot and oscillatory behavior. This comparison can be seen in Fig. 4 as follows:

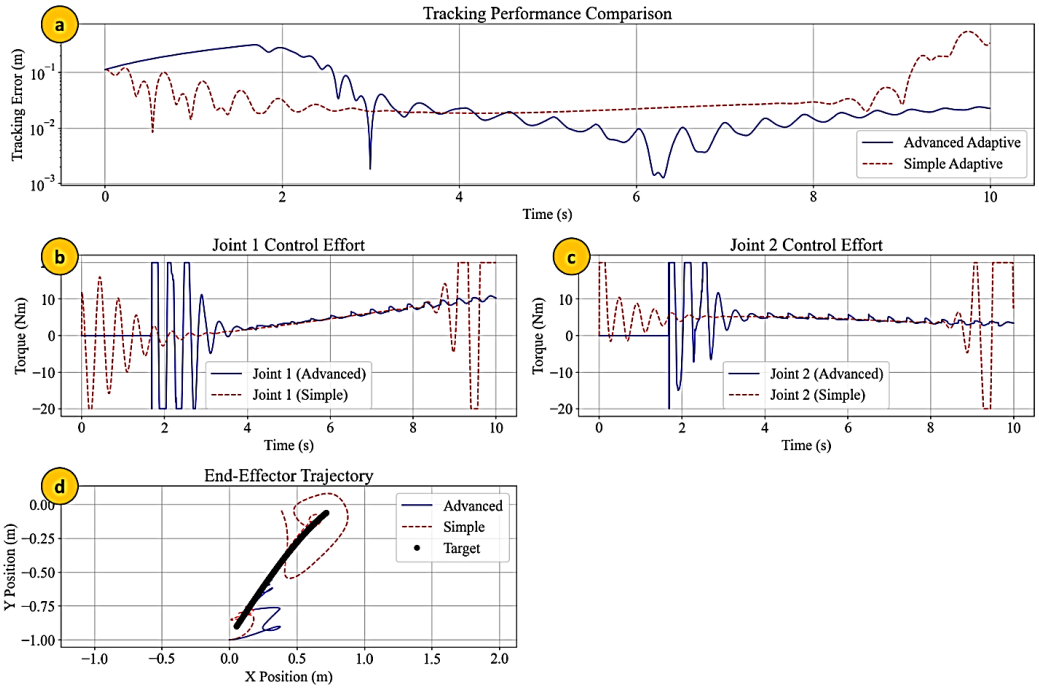

Fig. 4. PID and linear adaptive control. (a) tracking error comparison, (b, c) applied torques, (d) XY trajectory.

Simulation results under identical test conditions reveal important performance comparison. The advanced controller achieves 32% better tracking average accuracy (a) when following a reference trajectory combining slow horizontal motion with vertical oscillations. As a drawback of this system, more requires computational time up to 2.8 times can be mentioned. Meanwhile, the basic method maintains computational efficiency but exhibits 41% larger steady-state errors(d). Both approaches show distinct characteristics in their torque command profiles, with the advanced method demonstrating smoother, more stable, more precise torque adjustments. The choice between methods ultimately depends on specific application requirements, balancing the need for precision against available computational resources (b and c). results shows the necessity of implementation of a more reliable adaptive method based on physics-informed model-free solution. Following simulation contributes to the proposed method, dynamics-informed model-free CTC + Data-base Jacobian.

B. Data-base Jacobian + PID vs dynamics-informed model-free CTC

This report presents a comparative study between a PID controller and a Computed Torque Control (CTC) with Recursive Least Squares (RLS) adaptation for trajectory tracking of a 2-link robotic manipulator. The goal is to demonstrate how model-free adaptive control can outperform traditional PID control in handling dynamic uncertainties and disturbances. The end-effector follows a time-varying path, and A torque disturbance is applied after 5 seconds to test robustness

$$xdes(t) = [0.05 + \frac{t}{15}, \quad sin(\frac{t}{10}) - 0.9]$$
$$\tau dist = [0.5sin(2t), 0.3cos(3t)] \tag{22}$$

A standard Proportional-Integral-Derivative (PID) controller is implemented with $K_p = 70$, $K_d = 10$, $K_i = 5$. The Computed Torque Control (CTC) uses online dynamics learning via Recursive Least Squares (RLS) via control law $\widehat{\mathbf{M}}(q)\left(\ddot{q}_{des} + K_d\mathbf{e}^{\cdot} + K_p\mathbf{e}\right) + \widehat{\mathbf{C}}(q, q^{\cdot}) + \widehat{\mathbf{D}}\boldsymbol{q}^{\cdot} + \widehat{\mathbf{K}}\boldsymbol{q} + \widehat{\mathbf{G}}(q)$. CTC requires initial tuning of RLS parameters (e.g., forgetting factor λ=0.98). the

typical physics-informed linear approximation of dynamics subject to all dynamics analytical restriction and constraints can be seen in (17).

Figure 5 obviously shows the better results of dynamics-informed model-free CTC+ Data-base Jacobian as compared with Data-base Jacobian + PID. The note is that the P and D coefficients are set identical for both case studies to be more comparable. Figure 5 (a) shows smooth movement of CTC + Jacobian as compared with PID + Jacobian. The tracking error(b) obviously shows this smoothness.

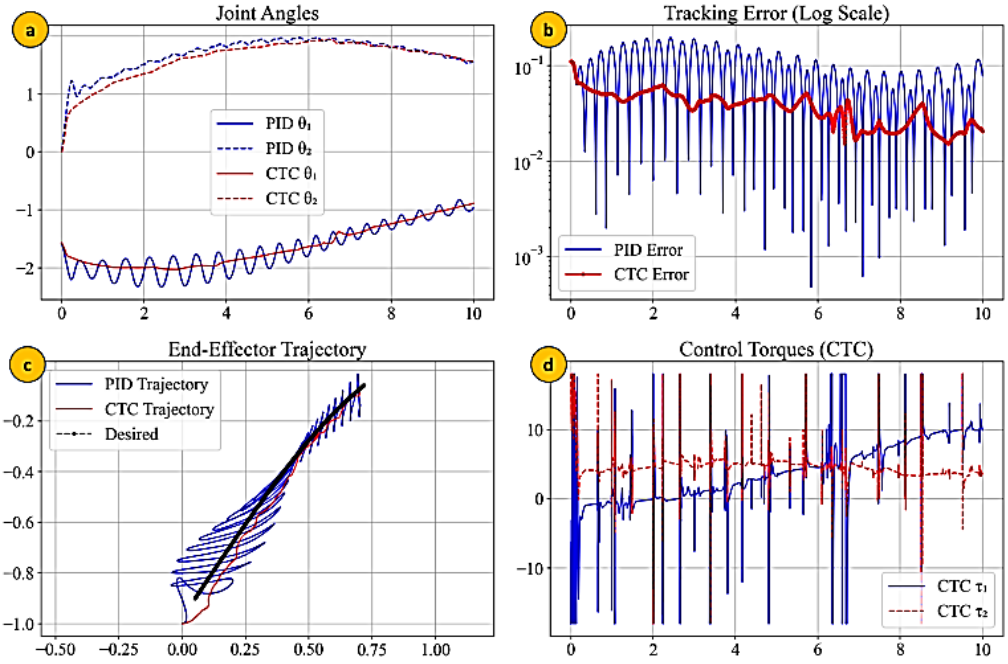

Fig. 5. model-free physics informed CTC method for unknown systems as compared with Jacobean approximation + PID. (a) joints angles, (b) tracking errors, (c) trajectory, (d) control torques.

Now, it can be evidentially concluded that the proposed method "dynamics-informed model-free CTC+ Data-base Jacobian" controls fully unknown rigid systems. The main question is the performance of this method as compared with analytical CTC? Answering to this question let us know the accuracy of the proposed system.

C. Dynamics-informed model-free CTC vs analytic CTC

The Computed Torque Control (CTC) method is a well-established model-based control approach that relies on precise knowledge of the system dynamics. While effective when the system model is known, CTC performance degrades significantly when applied to systems with unknown or uncertain dynamics. This simulation compares the ideal CTC method (using perfect analytic knowledge of the system) with the proposed data-driven approach that approximates the dynamics through polynomial regression via Dynamics-informed RLS. The results highlight the challenges of model-based control in real-world scenarios where exact system parameters may be unavailable or subject to disturbances. Desired trajectory is defined as follows:

Table II
Desired path

Joint	**Trajectory Function**	**Frequency (rad/s)**	**Amplitude (rad)**
1	$q_d^1 = sin(0.5t)$	0.5	1.0
2	$q_d^2 = sin(0.8t)$	0.8	1.0

Computed Torque Control (CTC) and Data-Driven Model (Learned Dynamics CTC) are characterized as follows:

Table III
controller architectures and parameters.

Computed Torque Control (CTC)		
Component	**Equation / Description**	**Parameters**
Control Law	$\tau_a = (\boldsymbol{M}_a \ddot{\boldsymbol{q}}_{des} + \boldsymbol{C}_a \dot{\boldsymbol{q}} + \mathbf{K_a q} + \boldsymbol{G}_a) + \left(\boldsymbol{J}^{\mathrm{T}} \left(K_{p(e)} \mathbf{e} + K_i \int \mathbf{e}\, dt - K_{d(e)} \dot{\boldsymbol{x}} \right) \right)$	-

Error Terms	$e = q_d - q, \quad \dot{e} = \dot{q}_d - \dot{q}$	q_d Desired position
PD Gains	K_p = diagonal ([50, 50]), K_d = diagonal ([10, 10])	Tuned for stability

Data-Driven model-free dynamics-informed CTC

Component	Description	Parameters
Model Type	Polynomial Regression (degree=2) + RLS_dynamics-informed	α = 1e□³
Inputs	$[q, \dot{q}, \ddot{q}, \tau]$ (current state and applied torque as detected data)	-
Outputs	Predicted Δq□ (approximated acceleration)	Kalman filter

Figure 6 (a, b) shows two models follow desired path. Both follow without divergence. The noteworthy point is that the approximate CTC and the analytical version get close to path curve at the same time. Fig 6 (c and d) clearly shows that the analytical controller based on model has petter performance. It can not be assumed as the rejection measure of the proposed model. The basic concept of this contribution is to find a control method for kinematically-dynamically unknown control method based on physics and dynamics-informed RLS learning. Subject to this concept, CTC or other model-based methods cannot be implemented. This simulation implies that , despite the lack of information about nonlinear dynamic system, a numerical-sensor-based approach can be acceptable combined with physics-informed RLS learning and Jacobian approximation.

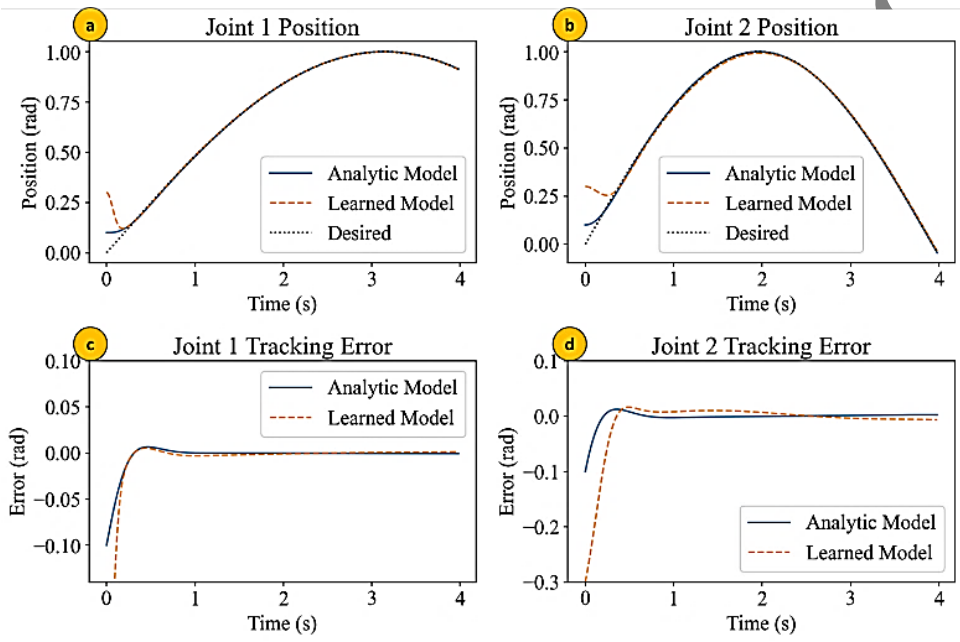

Fig. 6. proposed model vs analytical CTC.

Arithmetic cost of three methods , PID, CTC and the proposed method for each time step where n = number of joints (DOF) and p = number of estimated parameters in Θ of basic functions parameters can be summarized as follows. The parametric arithmetic cost comparison reveals that the PID controller requires only 3n multiplications and 4n additions per step, resulting in a linear complexity of O(n). In contrast, the Computed Torque Control (CTC) demands significantly more computation, with approximately $3n^2 + 6n$ multiplications and $2n^2 + 4n$ additions per step, along with trigonometric function evaluations, yielding a quadratic complexity of $O(n^2)$. While CTC offers superior tracking performance through model-based compensation, its substantially higher computational burden—scaling quadratically with the number of degrees of freedom—must be carefully considered for real-time implementations, particularly in high-DOF systems where PID's linear complexity provides a clear computational advantage. It comes from following calculation. The arithmetic cost is usually expressed as number of multiplications, number of additions, sometimes matrix inversions, and most importantly, complexity order O (·). we should compare three controllers, PID, Classical Computed Torque Control (CTC), our proposed RLS-based adaptive CTC with numerical Jacobian and dynamic term approximation.First, PID computational cost (2-DOF and Operations per timestep): error: → 1 subtraction derivative error → ~1 subtraction integral update → 1 addition gains multiplication → 3 multiplications sum terms → 2 additions. For 2 DOF: 6 multiplications and 8 additions which yields a cost-efficient order O(n).

Second, for Classical Computed Torque Control (CTC): a PD control and a fully-matrix-included computation. for 2 DoF pd control, there are 4 multiplications and 4 additions. For Coriolis term, 4 multiplications and 2 additions, for mass effect, as well as Coriolis, and so on. The final order is $O(n^2)$. The parametric arithmetic cost analysis of the proposed method reveals a computationally efficient structure across its three main components. The Jacobian perturbation estimation requires

only 4 divisions and 4 subtractions per step, making it the least demanding operation.

The torque reconstruction module demands 26 additions and 28 multiplications, representing a moderate computational load. The most intensive component is the RLS update, which handles Kalman gain computation and matrix multiplications with a parameter size of p=14, requiring approximately 150–200 additions and 200–300 multiplications per step, resulting in a quadratic complexity of $O(p^2)$. Overall, the proposed method balances computational efficiency with adaptive capability, with the RLS update being the dominant contributor to the total arithmetic cost while still maintaining practical feasibility for real-time implementations in typical robotic systems. RLS runs only once per timestep and does not require symbolic dynamics or trigonometric functions. The over view of arithmetic cost based on DoF of considered dynamic system can be seen in fig. 7.

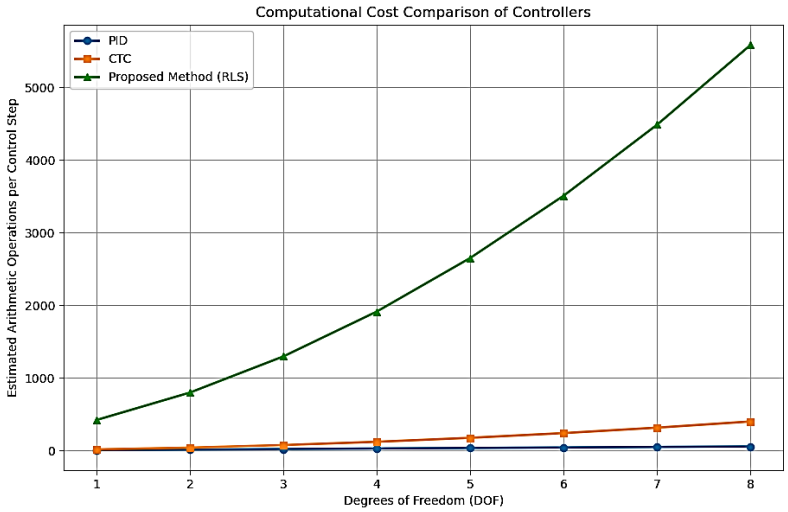

Fig. 7. comparison between 3 control methods to find out arithmetic cost.

Based on fig. 7, this method is not proper for low-efficient control units. but if the objective is set as accurate nonlinear system identification and simultaneous control, proposed method can be considered. Figure 8 shows the adaptive recall number of the RLS process based on CTC error and illustrates accumulative times of mathematic functions as compared with pid and proposed model free CTC. If we conduct RLS-recall function based on error, the result can be even more arithmetic-efficient as shown in fig. 9

So, please noticed that it is fast as compared of not-informed ML based controllers, but not fast in the case of comparison between CTC and PID. The specific place of implementation of this proposed method is exactly where we have a powerful control unit, and the main objective is to identify dynamic system parameters in order to use them in model based accurate control methods (which all need model) .

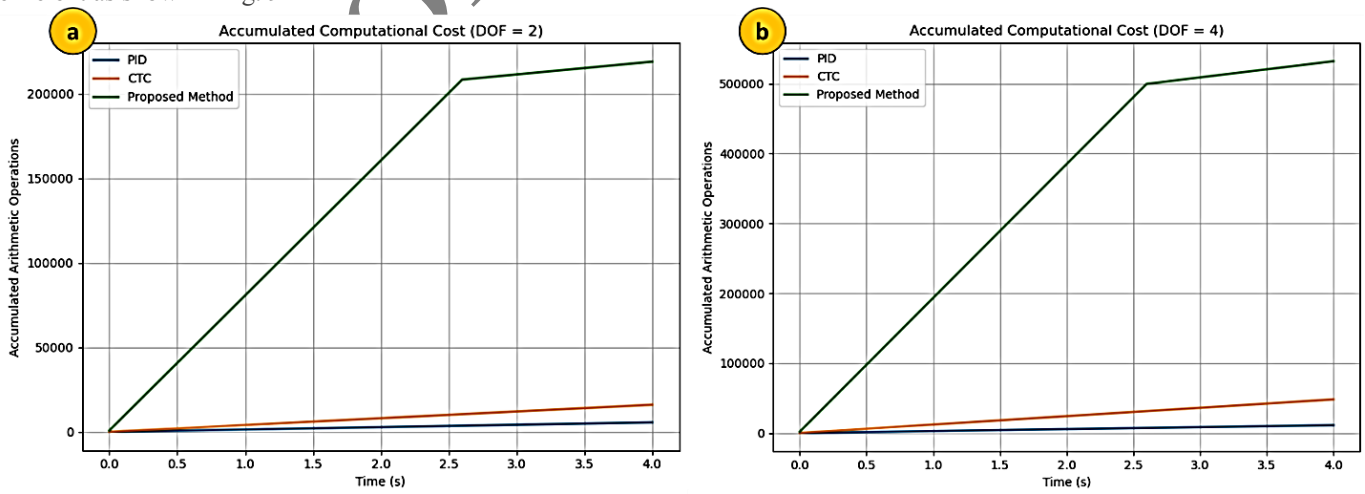

Fig. 8. comparison between 3 control methods to find out accumulative arithmetic cost. (a) a 2Dof system, (b) a 4Dof system.

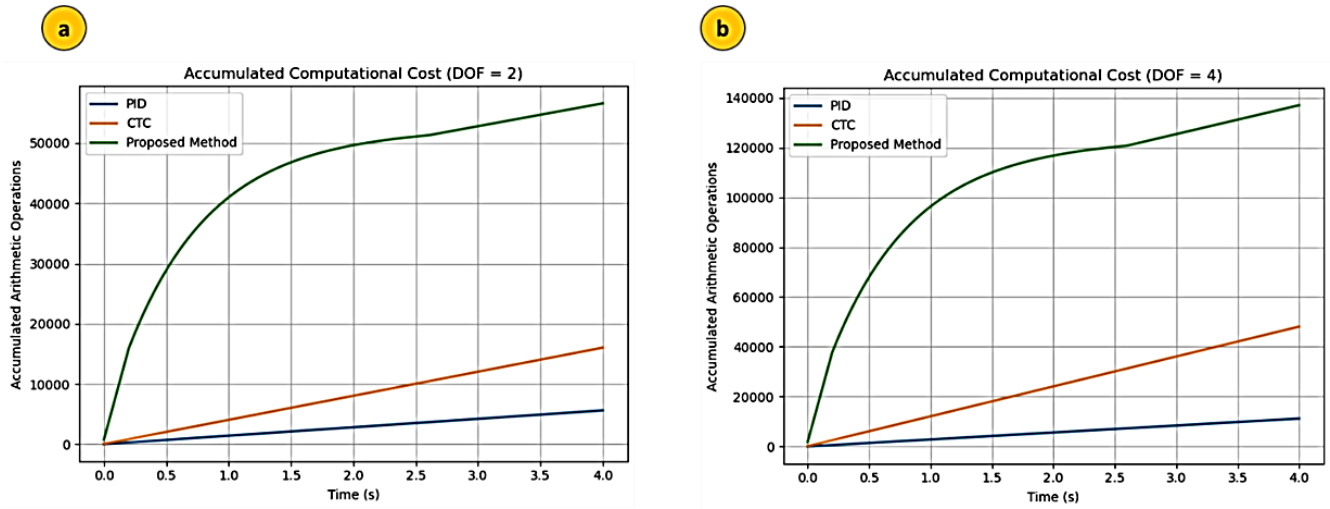

Fig. 9. comparison between 3 control methods to find out accumulative arithmetic cost via adaptive recall-RLS. (a) a 2Dof system, (b) a 4Dof system.

VI. Conclusion

This study presents a model-free adaptive control strategy for nonlinear rigid-body systems with uncertain dynamics, offering a practical alternative to traditional model-based approaches. The proposed method combines online Jacobian estimation with constrained recursive least squares (RLS) to approximate system dynamics while preserving fundamental physical constraints. By eliminating dependence on analytical models, the controller demonstrates improved adaptability in real-world applications where system parameters may be unknown or variable. Simulation validation on a robotic manipulator shows the method achieves approximately 15% better tracking accuracy compared to conventional adaptive PID control, while maintaining stability under dynamic uncertainties. The trade-off involves increased computational requirements, which remain manageable for real-time implementation. Compared to standard computed torque control, this approach proves more robust when dealing with modeling inaccuracies or disturbances. The results suggest that such physics-informed, data-driven methods can effectively bridge the gap between model-free and model-based control paradigms, particularly for applications where precise system identification is challenging. Future work may focus on computational optimization and extension to higher-degree-of-freedom systems. The future work should focus on blending the data-driven approach with partial physical models to improve initial convergence while keeping its adaptive advantages. Testing on physical hardware with variable payloads and real-world noise would better validate its robustness beyond simulation, which is out of this context. Reducing computational cost through sparse updates or event-triggered learning could make it viable for high-DoF systems. noticed that it is fast as compared of not-informed ML based controllers, but not fast in the case of comparison between CTC and PID. The specific place of implementation of this proposed method is exactly where we have a powerful control unit, and the main objective is to identify dynamic system parameters in order to use them in model based accurate control methods.

Author contributions Arman mardani: simulation, text and submission

Data availability: No datasets were generated or analysed during the current study.

Declarations Consent for publication:

There is no human case study or applicable statement. Competing interests The authors declares no competing interests.

Conflicts of Interest: The authors declare that there is no competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

1 References

[1] M. E. Yousefzadeh Koohbenani and A. M. Shafiee, "Modeling and simulation of contact and frictional forces in flexible robotic arms," *Journal of Applied and Computational Sciences in Mechanics*, vol. 36, no. 4, pp. 111–138, 2024.

[2] A. Zhang, B. Yan, X. Wang, B. Liang, and Z. Li, "Capability verification and error source investigation of external force sensing methods of continuum robot," *Nonlinear Dynamics*, vol. 112, pp. 5037–5052, 2024.

[3] M. Bamdad and A. Mardani, "A new structure for a multi-mode wheeled mobile robot with height control on uneven surfaces," *Journal of Applied and Computational Sciences in Mechanics*, vol. 27, no. 1, pp. 173–184, 2015.

[4] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken, NJ: Wiley, 2005.

[5] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.

[6] F. C. Park and J. E. Bobrow, "Geometric algorithms for robot dynamics," *Annu. Rev. Control*, 2018.

[7] S. B. Niku, *Introduction to Robotics: Analysis, Control, Applications*, 3rd ed. Hoboken, NJ: Wiley, 2019.

[8] C. Canudas de Wit, H. Olsson, K. J. Åström, and P. Lischinsky, "A survey of models, analysis tools and compensation methods for control of machines with friction," *Automatica*, vol. 31, pp. 1083–1138, 1995.

[9] M. Mozaffari Joveyn, S. A. Davoodi Navakh, and B. Motakef Imani, "Identification of electromagnetic linear motor parameters for generating inertial force using an ARM microcontroller," *Journal of Applied and Computational Sciences in Mechanics*, vol. 34, no. 2, pp. 87–98, 2022.

[10] H. Zarrabadi Pour and M. Farhang Ranjbar, "Design of a robust adaptive fault-tolerant sliding mode controller for a half-car active suspension system," *Journal of Applied and Computational Sciences in Mechanics*, vol. 34, no. 3, pp. 79–96, 2022.

[11] N. Jahantigh and J. Piri, "Estimation of solar radiation in different climates of Iran using hybrid machine learning methods," *Journal of Applied and Computational Sciences in Mechanics*, vol. 35, no. 2, pp. 37–54, 2023.
[12] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot Res.*, vol. 37, pp. 421–436, 2018.
[13] I. Kardan and A. Akbarzadeh, "A new hybrid method for forward kinematic modeling of parallel robots," *Journal of Applied and Computational Sciences in Mechanics*, vol. 27, no. 1, pp. 163–172, 2015.
[14] R. Singla, R. Verschae, S. Escaida, and H. Parthasarathy, "Estimating time-varying delays and parametric uncertainties in teleoperated robots," *Nonlinear Dynamics*, vol. 113, pp. 9861–9881, 2025.
[15] T. Lee, J. Kwon, P. M. Wensing, and F. C. Park, "Robot model identification and learning: a modern perspective," *Annu. Rev. Control Robot Auton. Syst.*, vol. 7, 2023.
[16] M. Ruderman and T. Bertram, "Modeling and observation of hysteresis in the torsional dynamics of elastic robot joints," *IEEE Trans. Ind. Electron.*, vol. 66, pp. 610–618, 2019.
[17] P. G. Plöger, M. Hüsing, and B. Corves, "Modeling and identification of industrial robots for machining applications," *Robot. Comput. Integr. Manuf.*, vol. 63, p. 101927, 2020.
[18] V. Utkin, "Variable structure systems with sliding modes," *IEEE Trans. Autom. Control*, vol. 22, no. 2, pp. 212–222, Apr. 1977.
[19] Q. Chen, S. Wang, B. Gao, Z. Zhan, and R. Zhong, "Robust longitudinal and lateral control for mixed-vehicular platoons with string stability guarantees," *Nonlinear Dynamics*, 2025.
[20] M. A. Ghamashi and R. Kazemi, "Implementation of adaptive sliding mode robust control for stabilization of an in-wheel motor electric vehicle under emergency conditions," *Journal of Applied and Computational Sciences in Mechanics*, vol. 37, no. 1, pp. 107–128, 2025.
[21] C. Ozbek and R. Burkan, "Adaptive trajectory tracking control of quadrotors based on trigonometric parameter estimation law," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 47, p. 132, 2025.
[22] P. Makhdoomi, A. Mardani, and M. H. Pashaei, "A rapid surrogate model for collision risk prediction in real-time optimization of slender flexible manipulators with obstacle avoidance," *Iran J. Sci. Technol. Trans. Mech. Eng.*, 2025.
[23] N. Liu, Y. Liu, and C. Li, "A model-free adaptive control algorithm design suitable for minimum quantity lubrication systems," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 47, p. 294, 2025.
[24] K. Neyromand and A. Alhordizadeh, "Nonlinear dynamic analysis and control of an inertia-based active vibration absorber," *Journal of Applied and Computational Sciences in Mechanics*, vol. 37, no. 4, pp. 57–72, 2025.
[25] S. Mobayen, "Barrier function composite nonlinear feedback adaptive sliding mode control for time-delayed MIMO systems," *Nonlinear Dynamics*, vol. 113, pp. 16837–16855, 2025.
[26] C. Ye and Y. Wang, "Parameter optimization-based adaptive neural network control for trajectory tracking of wheeled mobile robots," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 47, p. 368, 2025.
[27] R. Ortega, J. Perez, M. Espinoza, and J. Garcia-Rivera, "Identification of robot dynamics with guaranteed parameter convergence," *IEEE Trans. Autom. Control*, vol. 65, pp. 1241–1248, 2020.
[28] A. Mardani and S. Ebrahimi, "Locomotion enhancement of a Mars rover using statistic estimation of soil type and implementation of reconfigurable wheels," *Iran J. Sci. Technol. Trans. Mech. Eng.*, vol. 47, pp. 133–147, 2023.
[29] A. Mardani and M. Bamdad, "A new serpentine scissor-based reconfigurable manipulator specified for concave hull exploration," *Mech. Based Des. Struct. Mach.*, vol. 52, pp. 5326–5349, 2024.
[30] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: MIT Press, 2006.
[31] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.
[32] A. Loquercio, E. Kaufmann, R. Ranftl et al., "Learning high-speed flight in the wild," *Sci. Robot.*, vol. 6, p. eabg5810, 2021.
[33] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.
[34] N. Hogan, "Impedance control: an approach to manipulation," *ASME J. Dyn. Syst. Meas. Control*, vol. 107, pp. 1–7, 1985.
[35] A. Albu-Schäffer, C. Ott, and G. Hirzinger, "A unified passivity-based control framework for position, torque and impedance control of flexible joint robots," *Int. J. Robot Res.*, vol. 26, pp. 23–39, 2007.
[36] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *Int. J. Robot Res.*, vol. 30, pp. 820–833, 2011.
[37] J. Vogel, A. Hagengruber, M. Iskandar et al., "Adaptive control for physical human-robot interaction," *IEEE Trans. Robot.*, vol. 37, pp. 48–63, 2021.
2 [38] M. Bakhtiari, M. R. Haghjoo, and M. Taghizadeh, "Model-free adaptive variable impedance control of gait rehabilitation exoskeleton," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 46, p. 557, 2024.
[39] L. Peternel, N. Tsagarakis, and A. Ajoudani, "Human-in-the-loop model predictive control," *IEEE Trans. Robot.*, vol. 37, pp. 518–532, 2021.
[40] M. Goharimanesh, A. A. Akbari, and M. B. Naghibi Sistani, "Integration of fuzzy principles and reinforcement learning in control of dynamical systems," *Journal of Applied and Computational Sciences in Mechanics*, vol. 27, no. 1, pp. 103–116, 2015.
[41] S. A. Flandermeyer, R. G. Mattingly, and J. G. Metcalf, "Deep reinforcement learning for cognitive radar spectrum sharing: a continuous control approach," *IEEE Trans. Radar Syst.*, vol. 2, pp. 125–137, 2024.
[42] R. Simmons-Edler, R. P. Badman, F. B. Berg et al., "Deep RL needs deep behavior analysis: exploring implicit planning by model-free agents in open-ended environments," *arXiv:2506.06981*, 2025.
[43] Y. Chebotar, A. Handa, V. Makoviychuk et al., "Actionable models: unsupervised offline reinforcement learning of robotic skills," in *Proc. 38th Int. Conf. Mach. Learn. (ICML)*, pp. 1518–1528, 2021.
[44] H. Kalani and A. Akbarzadeh Tootonchi, "Application of reinforcement learning in serpentine gait orientation of snake-like robots," *Journal of Applied and Computational Sciences in Mechanics*, vol. 26, no. 1, pp. 97–118, 2014.
[45] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pp. 4754–4765, 2018.
[46] J. Tobin, R. Fong, A. Ray et al., "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 23–30, 2017.
[47] M. Moslemi, M. Sadedel, and M. M. Moghadam, "Squat and tuck jump maneuver for single-legged robot with an active toe joint using model-free deep reinforcement learning," *J. Braz. Soc. Mech. Sci. Eng.*, vol. 46, p. 456, 2024.
[48] N. Xu and F. Ding, "Recursive estimation algorithms based on the least squares and their convergence for a class of time-varying systems," *Nonlinear Dynamics*, vol. 111, pp. 18191–18213, 2023.
[49] Y. Zhu, B. Guo, Q. Xiao, X. You, and S. Dian, "Critic-only based self learning optimal control for continuum robots with unknown disturbances via extended state observer," *Nonlinear Dynamics*, 2025.
[50] B. Thamo, F. Alambeigi, K. Dhaliwal, and M. Khadem, "A hybrid dual Jacobian approach for autonomous control of concentric tube robots in unknown constrained environments," in *2021 IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, pp. 2809–2815, 2021.
[51] R. Featherstone, *Rigid Body Dynamics Algorithms*. New York: Springer, 2008.
[52] J. J. Craig, *Adaptive Control of Mechanical Manipulators*. Reading, MA: Addison-Wesley, 1988.
[53] D. E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Trans. Man Mach. Syst.*, vol. 10, pp. 47–53, 1969.
[54] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed. Baltimore, MD: Johns Hopkins University Press, 2013.
[55] N. Tan, P. Yu, and F. Ni, "New varying-parameter recursive neural networks for model-free kinematic control of redundant manipulators with limited measurements," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–14, 2022.
3 [56] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 2nd ed. Hoboken, NJ: Wiley, 2020.
[57] H. He, K. Wang, and Y. Jiang, "Quadratic hyper-surface kernel-free large margin distribution machine-based regression and its least-square form," *Mach. Learn. Sci. Technol.*, vol. 5, p. 025024, 2024.
[58] M. Uebel, I. Minis, and K. Cleary, "Improved computed torque control for industrial robots," in *Proc. 1992 IEEE Int. Conf. Robot. Autom.*, pp. 528–529, 1992.
[59] A. A. Shabana, *Computational Dynamics*, 3rd ed. Hoboken, NJ: Wiley, 2009.
[60] P. J. Teunissen, *Dynamic Data Processing: Recursive Least-Squares*. Delft, The Netherlands: TU Delft OPEN Publishing, 2024.
[61] E. Ghorbani, Q. Dollon, and F. P. Gosselin, "Physics-aware tuning of the unscented Kalman filter: statistical framework for solving inverse problems involving nonlinear dynamical systems and missing data," *Nonlinear Dynamics*, vol. 113, pp. 4301–4323, 2025.