Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Physics-informed neural networks for the prediction of robot dynamics considering motor and external force couplings*

Fengyu SUN, Shuangshuang WU, Zhiming LI, Peilin XIONG, Wenbai CHEN†‡

College of Automation, Beijing Information Science and Technology University, Beijing 100192, China

†E-mail: chenwb@bistu.edu.cn

Received Apr. 21, 2025; Revision accepted Sept. 28, 2025; Crosschecked Nov. 13, 2025; Published online Dec. 16, 2025

Abstract: In recent years, physics-informed neural networks (PINNs) have shown remarkable potential in modeling conservative systems of rigid-body dynamics. However, when applied to practical interaction tasks of manipulators (e.g., part assembly and medical operations), existing PINN frameworks lack effective external force modeling mechanisms, resulting in significantly degraded prediction accuracy in dynamic interaction scenarios. Additionally, because industrial robots (including UR5 and UR10e robots) are generally not equipped with joint torque sensors, obtaining precise dynamics training data remains challenging. To address these issues, this study proposes two enhanced PINNs that integrate motor dynamics and external force modeling. First, two data-driven Jacobian matrix estimation methods are introduced to incorporate external forces: one learns the mapping between end-effector velocity and joint velocity to approximate the Jacobian matrix, while the other first learns the system's kinematic behavior and then derives the Jacobian matrix through analytical differentiation of the forward kinematics model. Second, current-to-torque mapping is embedded as physical prior knowledge to establish direct correlations between system motion states and motor currents. Experimental results on two different manipulators demonstrate that both models achieve high-precision torque estimation in complex external force scenarios without requiring joint torque sensors. Compared with state-of-the-art methods, the proposed models improve overall modeling accuracy by 31.12% and 37.07% on average across various complex scenarios, while reducing joint trajectory tracking errors by 40.31% and 51.79%, respectively.

Key words: Dynamics modeling; Physics-informed neural networks; Motor dynamics; External force modeling; Kinematics

https://doi.org/10.1631/FITEE.2500254 **CLC number:** TP241.2

1 Introduction

With advancements in industrial automation and intelligent technologies, manipulators are increasingly deployed in complex dynamic environments, including precision assembly, minimally invasive surgery, and human–robot collaboration. These applications expose manipulators to significant environmental interaction forces (e.g., contact forces during assembly and reaction forces from surgical instruments) and inherent nonlinearities (e.g., joint

‡ Corresponding author

* Project supported by the Beijing Municipal Natural Science Foundation-Xiaomi Innovation Joint Fund (No. L233006), the National Natural Science Foundation of China (Nos. 62276028 and 92267110), the Qin Xin Talents Cultivation Program at Beijing Information Science and Technology University (No. QXTCP A202102), and the Beijing Information Science and Technology University School Research Fund (No. 2023XJJ12)

ORCID: Fengyu SUN, https://orcid.org/0009-0004-9992-6889; Wenbai CHEN, https://orcid.org/0000-0001-7683-2776

friction and transmission backlash). Such time-varying disturbances substantially modify the system's dynamic behavior (Phong et al., 2015). Conventional physics-based modeling methods (e.g., Lagrangian and Newton–Euler formulations) (Caccavale et al., 2003; Shah et al., 2012) often fail to capture these nonlinear dynamics due to their inherent simplifying assumptions and linearization approximations, resulting in notable discrepancies between predicted and actual system responses that degrade control precision and robustness (Sousa and Cortesão, 2014). Therefore, high-fidelity dynamic modeling under complex operating conditions is essential for precise manipulator control.

In recent years, purely data-driven methods have been widely applied in the field of dynamics modeling (Laddach et al., 2022; Akbari et al., 2023; Chen et al., 2025; Cheng et al., 2025; Pikuliński et al., 2025; Tao et al., 2025). These methods learn system dynamics directly from observed data, eliminating the dependence on system parameters inherent in traditional physics-based modeling, and demonstrate significant advantages when handling highly nonlinear, complex systems. However, their intrinsic black-box nature makes it difficult to reveal the underlying physical mechanisms of the system, which not only severely limits the understanding of system fundamentals and result interpretability (Duong et al., 2024), but also tends to cause overfitting issues due to the failure to capture the system's underlying structure. This may lead to predictions that violate actual physical behaviors in practical applications, thereby restricting their engineering applicability.

In recent years, physics-informed neural networks (PINNs) incorporating physical priors with deep learning have demonstrated remarkable advantages in robotic dynamics modeling (Roehrl et al., 2020; Djeumou et al., 2022; Xu et al., 2022; Lutter and Peters, 2023; Chrosniak et al., 2024; Gu et al., 2024; Li ZL et al., 2024; Trinh et al., 2025). PINNs integrate physical priors in deep learning frameworks using two approaches: first, by hard-coding physical laws directly into the network architecture to ensure outputs strictly satisfy physical constraints; second, by constructing physics-based regularization loss functions that guide the model to learn physically consistent behaviors through soft constraints. Compared to traditional purely data-driven methods, this physics–data fusion paradigm significantly enhances model generalization under data-scarce conditions while guaranteeing physical consistency in dynamics predictions.

A representative example is the deep Lagrangian networks (DeLaNs) proposed by Lutter et al. (2019), which encoded the Lagrangian mechanics framework into the neural network architecture while strictly preserving the symmetry and positive definiteness of the inertia matrix during learning. This approach achieved high-accuracy dynamics modeling while ensuring physical interpretability. Experimental results demonstrated that DeLaNs could precisely characterize the dynamics of conservative systems. However, due to the conservative assumptions inherent in the Lagrangian framework, they fail to accurately represent dynamics dominated by non-conservative forces (e.g., nonlinear friction and time-varying environmental disturbances), resulting in significantly degraded modeling accuracy when applied to practical complex dynamic engineering scenarios.

To address DeLaN's limitations in modeling non-conservative forces, Lutter et al. (2019) and Lutter and Peters (2023) integrated parameterized friction coefficients as trainable network weights into the architecture, achieving an organic fusion of frictional dynamics with the Lagrangian framework. Gupta et al. (2020) proposed combining the DeLaN architecture with a black-box dissipative force network, where DeLaN models conservative system dynamics while the dissipative network learns non-conservative forces. Lahoud et al. (2024) enhanced the modeling of complex friction phenomena by integrating the DeLaN framework with a linear asymmetric Coulomb friction model. Hu et al. (2025) developed an innovative hybrid learning framework that structurally combines Lagrangian dynamics with the Stribeck friction model and introduces a residual compensation module to capture unmodeled nonlinear dynamics. Liu et al. (2024) extended DeLaN by employing a black-box network to learn damping matrices for dissipative force modeling and proposed a method that incorporates dissipation while allowing non-collocated control actions. Wu et al. (2024) and Li ZM et al. (2024, 2025) effectively modeled non-conservative dynamics (including nonlinear friction, external disturbances, and elastic effects) that are difficult to characterize within traditional Lagrangian frameworks by fusing DeLaN with standard

deep networks.

Although these models demonstrate superior performance in handling complex friction phenomena, they lack effective external force modeling mechanisms and exhibit insufficient capability in characterizing interactive forces. This limitation leads to non-negligible dynamics modeling errors in practical force-sensitive applications (e.g., high-precision assembly and surgical instrument manipulation), ultimately compromising the accuracy and robustness of the control system. Furthermore, cost-constrained industrial manipulators are typically not equipped with joint torque sensors, making traditional dynamics learning methods that rely on joint torque measurements difficult to implement.

To address these challenges, this paper proposes two novel PINN-based dynamics models that incorporate motor dynamics and external force coupling for manipulators without joint torque sensors. First, we innovatively develop two data-driven Jacobian matrix estimation methods to integrate explicit external force representation mechanisms into the DeLaN architecture, enabling accurate modeling of nonlinear external forces in complex interaction scenarios: (1) the direct Jacobian method learns the mapping between end-effector velocity and joint velocity to estimate the Jacobian matrix; (2) the kinematic differentiation method first learns the system's kinematic behavior, and then analytically derives the Jacobian matrix through differentiation of the learned forward kinematics model.

Second, we encode motor dynamics into the DeLaN framework by incorporating current-to-torque mapping as a physical prior, establishing an explicit dynamic relationship between system motion states and motor currents for high-precision modeling without torque sensors, while simultaneously learning physically consistent friction model parameters end-to-end. Consequently, we develop two enhanced PINN-based dynamics models by integrating motor dynamics and external force modeling into the DeLaN framework: DeLaN with motor dynamics and kinematic differentiation force modeling (DeLaN-MKF) and DeLaN with motor dynamics and direct Jacobian force modeling (DeLaN-MJF). Extensive experiments conducted on various manipulators in both simulation and real-world environments demonstrate the effectiveness of our models compared to state-of-the-art approaches. The main contributions of this work are as follows:

1. Propose two novel PINN-based dynamics models that incorporate motor dynamics and external force coupling, which significantly improve dynamics modeling in complex interaction scenarios.

2. Develop two neural network-based Jacobian matrix estimation methods, where the kinematic differentiation method implicitly satisfies kinematic geometric constraints during training and exhibits stronger physical consistency and interpretability.

3. Embed motor dynamics into DeLaN to eliminate dependence on joint torque measurements while parameterizing friction coefficients as trainable network weights for end-to-end system dynamics learning.

4. Conduct extensive experiments on UR5 and UR10e manipulators across various complex interaction scenarios, demonstrating superior modeling accuracy, generalization capability, and control performance compared to state-of-the-art methods.

5. The DeLaN-MKF model implicitly satisfies geometric constraints during training and shows the best performance in multiple experiments, indicating that this method of improving physical interpretability by satisfying geometric constraints can further enhance model performance.

2 Preliminaries

2.1 Lagrangian mechanics

The core of Lagrangian mechanics lies in deriving dynamic equations using the Lagrangian, defined as $\boldsymbol{L}(\boldsymbol{q},\dot{\boldsymbol{q}})=\boldsymbol{T}(\boldsymbol{q},\dot{\boldsymbol{q}})-\boldsymbol{V}(\boldsymbol{q})$, where $\boldsymbol{T}(\boldsymbol{q},\dot{\boldsymbol{q}})=\frac{1}{2}\dot{\boldsymbol{q}}^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{q})\dot{\boldsymbol{q}}$ represents the kinetic energy with $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ denoting the generalized coordinates and the generalized velocities, respectively. $\boldsymbol{H}(\boldsymbol{q})$ is a symmetric positive-definite mass matrix ensuring positive kinetic energy for all nonzero velocities. $\boldsymbol{V}(\boldsymbol{q})$ represents the potential energy determined by the system's configuration. The Euler–Lagrange equation describing system dynamics derived using the calculus of variations is as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{L}(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\right)-\frac{\partial \boldsymbol{L}(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \boldsymbol{q}}=\boldsymbol{\tau}, \tag{1}$$

where $\boldsymbol{\tau}$ denotes the generalized force acting on the generalized coordinates $\boldsymbol{q}$ of the system. Substituting the kinetic energy $\boldsymbol{T}(\boldsymbol{q},\dot{\boldsymbol{q}})=\frac{1}{2}\dot{\boldsymbol{q}}^{\mathrm{T}}\boldsymbol{H}(\boldsymbol{q})\dot{\boldsymbol{q}}$ and potential energy $\boldsymbol{V}(\boldsymbol{q})$ into the Euler–Lagrange

equation yields the second-order ordinary differential equation that describes any holonomic multi-particle mechanical system:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\underbrace{\dot{\boldsymbol{H}}(\boldsymbol{q})\dot{\boldsymbol{q}}-\frac{1}{2}\left(\dot{\boldsymbol{q}}^{\mathrm{T}}\frac{\partial \boldsymbol{H}(\boldsymbol{q})}{\partial \boldsymbol{q}}\dot{\boldsymbol{q}}\right)^{\mathrm{T}}}_{\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})}+\underbrace{\frac{\partial \boldsymbol{V}(\boldsymbol{q})}{\partial \boldsymbol{q}}}_{\boldsymbol{g}(\boldsymbol{q})}=\boldsymbol{\tau}, \tag{2}$$

where $\ddot{\boldsymbol{q}}$ denotes the generalized accelerations, $\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})$ represents the Coriolis and centrifugal force matrix, and $\boldsymbol{g}(\boldsymbol{q})$ denotes the gravity vector. The equation simplifies to the standard rigid-body dynamics formulation for robotic systems:

$$\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}}+\boldsymbol{c}(\boldsymbol{q},\dot{\boldsymbol{q}})+\boldsymbol{g}(\boldsymbol{q})=\boldsymbol{\tau}. \tag{3}$$

2.2 DeLaN

The DeLaN is a neural network architecture based on Lagrangian mechanics (Lutter et al., 2019), incorporating Lagrangian mechanics as physical priors into the network structure to ensure that the learned dynamical models rigorously comply with physical constraints.

The network architecture of DeLaN is illustrated in Fig. 1. The framework independently parameterizes the mass matrix $\boldsymbol{H}(\boldsymbol{q})$ and potential energy $\boldsymbol{V}(\boldsymbol{q})$ to learn system dynamics. Given the symmetric positive-definite property of $\boldsymbol{H}(\boldsymbol{q})$, the kinetic energy remains positive for all nonzero velocities. To preserve this physical constraint, DeLaN employs Cholesky decomposition to represent the mass matrix as a product of lower triangular matrices:

$$\hat{\boldsymbol{H}}(\boldsymbol{q};\theta)=\hat{\boldsymbol{L}}(\boldsymbol{q};\theta)\hat{\boldsymbol{L}}^{\mathrm{T}}(\boldsymbol{q};\theta), \tag{4}$$

where $\hat{\boldsymbol{L}}(\boldsymbol{q};\theta)$ is a lower triangular matrix with positive diagonal elements. The off-diagonal elements $\hat{\boldsymbol{l}}_{\mathrm{o}}(\boldsymbol{q};\theta)$ use linear activation functions, while the diagonal elements $\hat{\boldsymbol{l}}_{\mathrm{d}}(\boldsymbol{q};\theta)$ employ non-negative activation functions plus an additional positive scalar b to ensure the diagonal positivity of $\hat{\boldsymbol{L}}(\boldsymbol{q};\theta)$. This decomposition strictly enforces the symmetry and positive definiteness of the mass matrix.

The optimization problem for DeLaN is formulated as the minimization of the residual of the Euler–Lagrange equation, as presented in Eq. (1):

$$\theta^*,\phi^*=\arg\min_{\theta,\phi}\left\|\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial \boldsymbol{L}(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\right)-\frac{\partial \boldsymbol{L}(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial \boldsymbol{q}}-\boldsymbol{\tau}\right\|^2_{W_\tau}, \tag{5}$$

where θ and ϕ are the trainable weight parameters for learning the mass matrix $\boldsymbol{H}(\boldsymbol{q})$ and potenial energy $\boldsymbol{V}(\boldsymbol{q})$, respectively; their optimal counterparts, θ^* and ϕ^*, are the values obtained from the optimization process. The norm $\|\cdot\|_{W_\tau}$ represents the diagonal covariance matrix of joint torques $\boldsymbol{\tau}$, accounting for residual magnitude variations across joints through gradient-based optimization. DeLaN combines data-driven learning with physical priors, providing enhanced physical interpretability over conventional neural networks.

3 Methodology

To overcome the limitations of DeLaN in modeling non-conservative forces (e.g., friction and external contact forces) that cannot be directly described

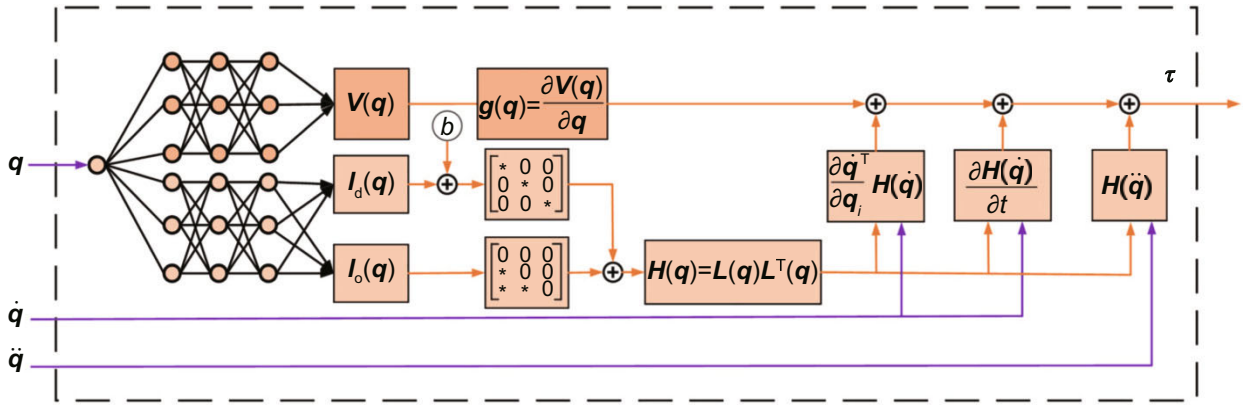


Fig. 1 Network architecture of DeLaN. The DeLaN architecture learns system dynamics through two dedicated neural networks that approximate the mass matrix $H(q)$ and the potential energy $V(q)$, respectively. DeLaN: deep Lagrangian network

by traditional Lagrangian frameworks, and to address the absence of joint torque measurements, this paper incorporates both the motor dynamics modeling and the external force modeling into DeLaN, proposing two PINN-based dynamics models.

To apply the general Lagrangian framework introduced in Section 2 to our specific robotic manipulator, we instantiate the generalized variables with their physical counterparts: the generalized coordinates $\boldsymbol{q}$ correspond to the joint angles, while their derivatives $\dot{\boldsymbol{q}}$ and $\ddot{\boldsymbol{q}}$ represent the joint velocities and accelerations, respectively. The generalized forces $\boldsymbol{\tau}$ are identified with the torques applied at each joint.

3.1 Motor dynamics modeling

Industrial manipulators are typically not equipped with joint torque sensors due to cost constraints. However, learning-based dynamics modeling methods rely on accurate joint torque data, thus motivating the proposed joint motor dynamics modeling approach for manipulators. The dynamics of direct current motors with harmonic reducers are generally described by the following equation:

$$\boldsymbol{\tau} = r k_{\mathrm{t}} \boldsymbol{i}, \tag{6}$$

where $\boldsymbol{i}$ represents the motor current, k_{t} denotes the motor torque constant, and r is the gear ratio. The joint torque is calculated from the motor current to establish a direct mapping relationship between the system's motion states and motor current. Due to the high-gear-ratio harmonic drives employed in the UR10e robot, which significantly amplify friction effects, it is essential to incorporate an accurate friction model to compensate for the nonlinear influences introduced by the harmonic drive.

In the UR10e robot, the friction in well-designed and maintained brushless motors is negligible, making harmonic drives the primary source of system friction. This study adopts the friction model from Clochiatti et al. (2024), originally developed for the UR5e robot, as both robots share similar motor configurations. The model assumes that the friction torque consists of a standard friction component and an asymmetric smoothed Coulomb friction component. The friction torque for the j^{th} motor is described by

$$\boldsymbol{\tau}_{\mathrm{fric},j} = k_{\mathrm{v}} \dot{\boldsymbol{\theta}}_{\mathrm{m}} + \tanh\left(\frac{\dot{\boldsymbol{\theta}}_{\mathrm{m}}}{\epsilon}\right) T_{\mathrm{c}}, \tag{7}$$

where the smoothing parameter ϵ is empirically set to 10^{-3}, $\dot{\boldsymbol{\theta}}_{\mathrm{m}}$ denotes the motor angular velocity, and k_{v} represents the friction coefficient. The Coulomb friction term T_{c} equals $T_{\mathrm{c}}^{\mathrm{dir}}$ for motor-to-load power flow and $T_{\mathrm{c}}^{\mathrm{rev}}$ for reversed flow. Here, $\tanh(\cdot)$ serves as a smooth transition function to approximate the discontinuous Coulomb friction, ensuring model differentiability across all velocities.

Because friction in the UR10e robot originates predominantly from harmonic drives, with the motor angular velocity $\dot{\boldsymbol{\theta}}_{\mathrm{m}} = r\dot{\boldsymbol{q}}$, the joint friction torque becomes

$$\boldsymbol{\tau}_{\mathrm{F}} = r\left(r k_{\mathrm{v}} \dot{\boldsymbol{q}} + \tanh\left(\frac{r\dot{\boldsymbol{q}}}{\epsilon}\right) T_{\mathrm{c}}\right), \tag{8}$$

$$T_{\mathrm{c}} = \begin{cases} T_{\mathrm{c}}^{\mathrm{dir}}, & \text{if } \dot{\boldsymbol{q}} \cdot \boldsymbol{\tau}_j \geq 0, \\ T_{\mathrm{c}}^{\mathrm{rev}}, & \text{if } \dot{\boldsymbol{q}} \cdot \boldsymbol{\tau}_j < 0. \end{cases} \tag{9}$$

The motor actuator model is incorporated into DeLaN, where the gear ratio r, motor torque constant k_{t}, Coulomb term T_{c}, and friction coefficient k_{v} are learned as network weights. Through end-to-end training, we achieve joint optimization of both the model and its parameters, ultimately establishing the dynamic mapping relationship between the system's motion states and motor currents. Unlike traditional parameter identification, these parameters are not calibrated through offline experiments but rather adapt dynamically as part of the modeling framework.

3.2 External force modeling

During grasping, assembly, and surgical operations, the robot end-effector inevitably contacts objects, generating external contact forces. Accurate modeling of these forces is essential for enhancing robotic performance and safety. The external torque formulation follows (Khatib, 1987):

$$\boldsymbol{\tau}_{\mathrm{ext}} = \boldsymbol{J}^{\mathrm{T}}(\boldsymbol{q}) \boldsymbol{F}_{\mathrm{ext}}, \tag{10}$$

where $\boldsymbol{\tau}_{\mathrm{ext}}$ denotes the external joint torque acting on each manipulator joint, $\boldsymbol{F}_{\mathrm{ext}}$ represents the end-effector external force/torque (measured by a six-axis force/torque sensor with components F_x, F_y, F_z, M_x, M_y, M_z), and $\boldsymbol{J}(\boldsymbol{q})$ is the Jacobian matrix mapping $\boldsymbol{F}_{\mathrm{ext}}$ to the joint-space torque $\boldsymbol{\tau}_{\mathrm{ext}}$. Due to undisclosed manufacturer parameters, obtaining an accurate $\boldsymbol{J}(\boldsymbol{q})$ proves challenging. This study develops two data-driven methods for Jacobian matrix approximation.

3.2.1 Direct Jacobian

The robot configuration is determined by its rigid links and joint angles $\boldsymbol{q}$. Through forward kinematics, the joint angles $\boldsymbol{q}$ are transformed into the end-effector pose $\boldsymbol{x}$ via $\boldsymbol{x} = f(\boldsymbol{q})$. The end-effector pose is defined as $\boldsymbol{x} = [p_x, p_y, p_z, \alpha, \beta, \gamma]^{\mathrm{T}}$, where p_x, p_y, and p_z denote the Cartesian position coordinates and α, β, and γ represent the orientation parameterized by ZYX Euler angles (roll–pitch–yaw sequence) (Crane III and Duffy, 1998). The function $f(\boldsymbol{q})$ is obtained through chained multiplication of homogeneous transformation matrices, representing the coordinate transformation from the base frame to the end-effector frame.

To analyze the relationship between end-effector velocity and joint velocity, the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$ is introduced and defined as

$$\dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{11}$$

where $\dot{\boldsymbol{x}}$ represents the end-effector velocity, defined as $\dot{\boldsymbol{x}} = [\boldsymbol{v}, \boldsymbol{\omega}]^{\mathrm{T}}$, with $\boldsymbol{v} = [v_x, v_y, v_z]^{\mathrm{T}}$ being the linear velocity vector (translational motion) and $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^{\mathrm{T}}$ the angular velocity vector (rotational motion). The Jacobian matrix establishes a differential mapping between joint space and task space. The direct Jacobian method directly learns the mapping between end-effector velocity and joint velocity. The network takes joint angles $\boldsymbol{q}$ as input and outputs the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$. The loss function is designed as follows:

$$L = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\dot{\boldsymbol{x}}}_i - \dot{\boldsymbol{x}}_i\right\|^2, \tag{12}$$

where N denotes the number of training samples and $\hat{\dot{\boldsymbol{x}}}$ indicates the predicted end-effector velocity from the model, computed as

$$\hat{\dot{\boldsymbol{x}}} = \hat{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{13}$$

where $\hat{\boldsymbol{J}}(\boldsymbol{q})$ denotes the Jacobian matrix predicted by the network. This approach essentially performs data-driven local linearization of the chain structure. However, this approach cannot globally model nonlinear kinematics. When training data only cover a local region of the workspace, significant prediction errors may occur in untrained areas. Moreover, kinematic geometric constraints may be violated due to data noise and overfitting.

3.2.2 Kinematic differentiation

The kinematic differentiation method first learns the system's kinematic behavior by fitting the forward kinematics model $\boldsymbol{x} = f(\boldsymbol{q})$, which captures the global nonlinear characteristics of the kinematics across the entire workspace. The Jacobian matrix is then obtained by computing the partial derivatives of the forward kinematics model $f(\boldsymbol{q})$ with respect to $\boldsymbol{q}$:

$$\boldsymbol{J}(\boldsymbol{q}) = \frac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial f_1}{\partial q_1} & \frac{\partial f_1}{\partial q_2} & \cdots & \frac{\partial f_1}{\partial q_n} \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} & \cdots & \frac{\partial f_2}{\partial q_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial q_1} & \frac{\partial f_m}{\partial q_2} & \cdots & \frac{\partial f_m}{\partial q_n} \end{bmatrix}. \tag{14}$$

The network takes joint angles $\boldsymbol{q}$ as input and outputs end-effector pose $\boldsymbol{x}$, trained through supervised learning. The loss function is designed to minimize the mean squared error between the predicted end-effector pose $\hat{\boldsymbol{x}}$ and the ground truth pose $\boldsymbol{x}$

$$L = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{\boldsymbol{x}}_i - \boldsymbol{x}_i\right\|^2. \tag{15}$$

The forward kinematics model fundamentally represents the chain rule of coordinate transformations for manipulator links. During the learning process, the network implicitly satisfies these geometric constraints imposed by the manipulator's structure (e.g., orthogonality of rotation matrices and invariance of link lengths) through training. As an analytical derivative of $f(\boldsymbol{q})$, each partial derivative $\frac{\partial f_i}{\partial q_j}$ in the Jacobian matrix inherits these geometric constraints, resulting in learned Jacobian matrices with strong physical consistency.

The Jacobian matrix $\boldsymbol{J}(\boldsymbol{q}) = \frac{\partial f(\boldsymbol{q})}{\partial \boldsymbol{q}}$ derived by this method maintains clear kinematic interpretations for each element $\frac{\partial f_i}{\partial q_j}$. This provides an intuitive understanding of how variations in joint angles affect the end-effector's position and orientation, directly revealing the spatial coupling relationships among the manipulator configuration parameters. The resulting Jacobian matrix exhibits high physical consistency and interpretability, enabling seamless integration of external force modeling into PINNs.

3.3 Design of the overall network architecture

Building upon Eq. (3) and considering friction effects and external disturbances, the system's

dynamic equation can be expressed as

$$\boldsymbol{\tau} = \boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_{\mathrm{F}} + \boldsymbol{\tau}_{\mathrm{ext}}, \quad (16)$$

where the joint output torque $\boldsymbol{\tau}$ includes conservative force terms (inertial, Coriolis, centrifugal, and gravitational forces) and non-conservative force terms (friction torque and external torque). After incorporating the motor dynamics model from Eq. (6), we obtain the final inverse dynamics equation for the manipulator:

$$\boldsymbol{i} = \frac{1}{rk_{\mathrm{t}}} \underbrace{\left(\boldsymbol{H}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau}_{\mathrm{F}} + \boldsymbol{\tau}_{\mathrm{ext}}\right)}_{\boldsymbol{\tau}}. \quad (17)$$

Fig. 2 shows the network architectures of both models (DeLaN-MJF and DeLaN-MKF). The models receive multiple inputs: the joint angles $\boldsymbol{q}$, angular velocities $\dot{\boldsymbol{q}}$, angular accelerations $\ddot{\boldsymbol{q}}$, and the six-dimensional end-effector external force $\boldsymbol{F}_{\mathrm{ext}}$. The network output is the motor current $\boldsymbol{i}$.

The orange portion represents the DeLaN framework, where we leverage DeLaN's advantages in conservative force modeling by using two separate networks to parameterize the mass matrix $\boldsymbol{H}(\boldsymbol{q})$ and potential energy $\boldsymbol{V}(\boldsymbol{q})$, thereby learning the system's conservative force terms, including inertial, Coriolis, centrifugal, and gravitational forces. The mass matrix $\boldsymbol{H}(\boldsymbol{q})$ is decomposed into a lower triangular matrix to ensure its symmetry and positive definiteness.

The blue portion represents the external force modeling, where two neural network methods are employed to fit the system's Jacobian matrix for mapping task-space forces to joint space, thus introducing an explicit external force modeling mechanism. The DeLaN-MKF model is based on kinematic differentiation: it first learns the system's kinematic behavior through a neural network that maps joint angles to end-effector pose ($\boldsymbol{x} = f(\boldsymbol{q})$), thereby obtaining the forward kinematics model $f(\boldsymbol{q})$. The Jacobian matrix is then derived by computing the partial derivatives of $f(\boldsymbol{q})$ with respect to $\boldsymbol{q}$. The DeLaN-MJF model adopts the direct Jacobian method, where a neural network directly learns the mapping from joint angular velocities to end-effector velocities to obtain the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$. Through the learned Jacobian matrix, the external torque term $\boldsymbol{\tau}_{\mathrm{ext}}$ is incorporated.

The purple portion represents the motor dynamics modeling. First, a friction model is embedded in the network: $\boldsymbol{\tau}_{\mathrm{F}} = r\left(rk_{\mathrm{v}}\dot{\boldsymbol{q}} + \tanh\left(\frac{r\dot{\boldsymbol{q}}}{\epsilon}\right)T_{\mathrm{c}}\right)$ to learn the system's nonlinear friction effects. Second, the mapping relationship between motor current and joint output torque is incorporated as physical prior knowledge, establishing a dynamic mapping between system motion states and motor currents, thereby eliminating dependence on joint torque measurements and achieving accurate modeling without torque sensors. Simultaneously, friction coefficients and motor dynamics parameters are learned

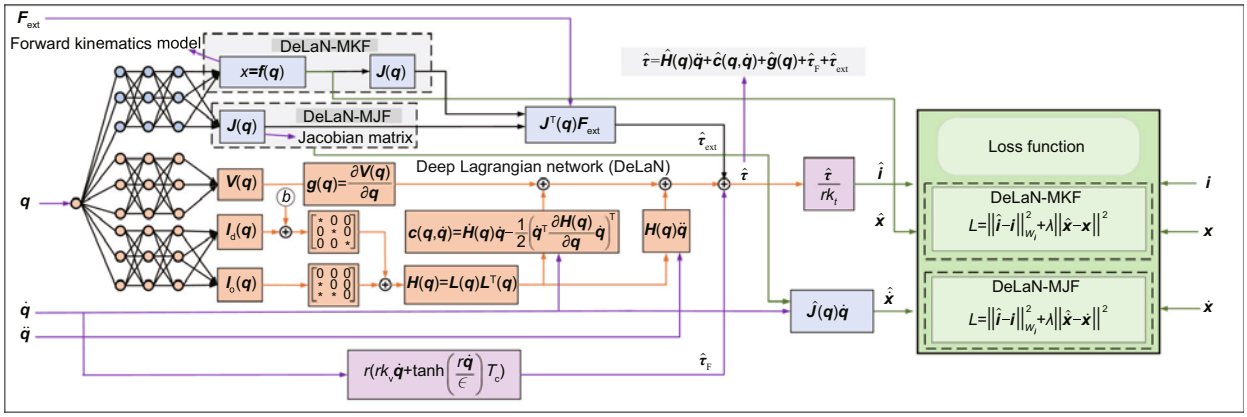


Fig. 2 Network architectures of two hybrid PINN-based models that incorporate motor dynamics and external force modeling into the DeLaN network. The orange components represent the DeLaN network, the blue components represent the external force modeling, the purple components represent the motor dynamics modeling, and the green components represent the loss function. DeLaN-MJF: DeLaN with motor dynamics and direct Jacobian force modeling; DeLaN-MKF: DeLaN with motor dynamics and kinematic differentiation force modeling; PINN: physics-informed neural network. References to color refer to the online version of this figure

as network weights through end-to-end training for joint optimization of the model and parameters.

The green portion shows the loss functions for both models. For DeLaN-MKF, the loss function is defined as $L = \left\|\hat{\boldsymbol{i}} - \boldsymbol{i}\right\|_{W_{\boldsymbol{i}}}^2 + \lambda \left\|\hat{\boldsymbol{x}} - \boldsymbol{x}\right\|^2$, where $\boldsymbol{i}$ represents the measured motor current, $\hat{\boldsymbol{i}}$ is the predicted motor current, including both dynamics loss (mean squared error between predicted current $\hat{\boldsymbol{i}}$ and measured current $\boldsymbol{i}$) and kinematics loss (mean squared error between predicted end-effector pose $\hat{\boldsymbol{x}}$ and actual pose $\boldsymbol{x}$) when fitting the Jacobian matrix. For DeLaN-MJF, the loss function is defined as $L = \left\|\hat{\boldsymbol{i}} - \boldsymbol{i}\right\|_{W_{\boldsymbol{i}}}^2 + \lambda \left\|\hat{\dot{\boldsymbol{x}}} - \dot{\boldsymbol{x}}\right\|^2$, which similarly comprises dynamics loss and Jacobian fitting loss (mean squared error between predicted end-effector velocity $\hat{\dot{\boldsymbol{x}}}$ and actual measured velocity $\dot{\boldsymbol{x}}$). The penalty term λ balances the relative weighting of these components in the total loss.

Overall, both models use the DeLaN architecture to learn the conservative force terms of the system. They establish explicit external force models by fitting the Jacobian matrix with different neural networks to learn the system's external torque. They introduce a friction model by parameterizing the friction coefficient as a trainable network weight integrated into the architecture, learning the friction torque. Therefore, the predicted joint output torque consists of three components: conservative force terms, friction torque, and external torque. Additionally, by embedding the mapping between motor current and joint output torque, a dynamic mapping between the system's motion state and motor current is ultimately established. The training process not only learns the system dynamics but also the Jacobian matrix. Hence, the loss functions of both models include two parts: the dynamics loss and the Jacobian matrix fitting loss.

DeLaN-MKF introduces the motor dynamics modeling and the external force modeling based on the kinematic differential method into the DeLaN architecture, while DeLaN-MJF introduces the motor dynamics modeling and the external force modeling based on the direct Jacobian method into the DeLaN architecture. In addition to the two proposed dynamic models, which simultaneously consider the coupling between motor dynamics and external forces, three other PINN-based dynamics models are constructed: (1) DeLaN-motor, which introduces only the motor dynamics modeling into the DeLaN architecture; (2) DeLaN-KF, which introduces only the external force modeling based on the kinematic differential method; (3) DeLaN-JF, which introduces only the external force modeling based on the direct Jacobian method. These five models are compared with the baseline model DeLaN to validate the effectiveness of the proposed motor dynamics method and the two external force modeling methods based on the learning of the Jacobian matrix.

3.4 Construction of the parametric network

The specific network parameterization of the models is described as follows:

$$\hat{\boldsymbol{i}} = \hat{f}^{-1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{F}_{\text{ext}}; \theta, \phi, \psi, k), \tag{18}$$

where ψ denotes the trainable weight parameter for learning the Jacobian matrix $\boldsymbol{J}(\boldsymbol{q})$. The parameter set k encompasses motor dynamics modeling parameters: reduction ratio r, torque constant k_{t}, Coulomb friction T_{c}, and friction coefficient k_{v}. The manipulator's inverse dynamics model is as follows:

$$\begin{aligned}\hat{f}^{-1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{F}_{\text{ext}}; \theta, \phi, \psi, k) = \frac{1}{r k_{\text{t}}} \Bigg(& \hat{\boldsymbol{H}}(\boldsymbol{q}; \theta) \ddot{\boldsymbol{q}} \\ & + \underbrace{\frac{\partial \hat{\boldsymbol{V}}(\boldsymbol{q}; \phi)}{\partial \boldsymbol{q}}}_{\hat{\boldsymbol{g}}(\boldsymbol{q}; \phi)} + \underbrace{\hat{\dot{\boldsymbol{H}}}(\boldsymbol{q}; \theta) \dot{\boldsymbol{q}} - \frac{1}{2} \left(\dot{\boldsymbol{q}}^{\mathrm{T}} \frac{\partial \hat{\boldsymbol{H}}(\boldsymbol{q}; \theta)}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \right)^{\mathrm{T}}}_{\hat{\boldsymbol{c}}(\boldsymbol{q}, \dot{\boldsymbol{q}}; \theta)} \\ & + \hat{\boldsymbol{\tau}}_{\text{F}}(\dot{\boldsymbol{q}}; k_{\text{v}}, T_{\text{c}}) + \hat{\boldsymbol{\tau}}_{\text{ext}}(\boldsymbol{q}, \boldsymbol{F}_{\text{ext}}; \psi) \Bigg).\end{aligned} \tag{19}$$

The training process learns both the system dynamics and the Jacobian matrix. To prevent the Jacobian matrix learning from dominating the training process at the expense of system dynamics learning, an additional penalty term is introduced. The optimization objective for DeLaN-MKF is defined as

$$\begin{aligned}(\theta^*, \phi^*, \psi^*, k^*) = \arg\min_{\theta, \phi, \psi, k} \Big(& \Big\| \hat{f}^{-1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{F}_{\text{ext}}; \\ & \theta, \phi, \psi, k) - \boldsymbol{i} \Big\|_{W_{\boldsymbol{i}}}^2 + \lambda \left\| \hat{\boldsymbol{x}}(\boldsymbol{q}; \psi) - \boldsymbol{x} \right\|^2 \Big).\end{aligned} \tag{20}$$

The optimization objective for DeLaN-MJF is defined as

$$\begin{aligned}(\theta^*, \phi^*, \psi^*, k^*) = \arg\min_{\theta, \phi, \psi, k} \Big(& \Big\| \hat{f}^{-1}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \boldsymbol{F}_{\text{ext}}; \\ & \theta, \phi, \psi, k) - \boldsymbol{i} \Big\|_{W_{\boldsymbol{i}}}^2 + \lambda \left\| \hat{\dot{\boldsymbol{x}}}(\boldsymbol{q}; \psi) - \dot{\boldsymbol{x}} \right\|^2 \Big).\end{aligned} \tag{21}$$

The primary learning objective focuses on system dynamics, requiring balanced optimization between dynamics loss and Jacobian matrix loss. This approach maximizes system dynamics learning while preserving essential Jacobian matrix accuracy.

4 Experiments

Multiple experiments are conducted on six-degree-of-freedom (6-DOF) UR5 and UR10e robots to validate the effectiveness of the proposed model. These two types of 6-DOF manipulators have joints numbered sequentially from the base (Joint 0) to the end-effector (Joint 5). Joint 5 is the joint that drives the end-effector.

4.1 Experimental setup

4.1.1 Experimental design

1. Motor dynamics modeling experiment

To verify the effectiveness of the proposed motor dynamics modeling method, inverse dynamics modeling experiments are conducted on the UR10e robot (without joint torque sensors) in both an ideal simulation platform and a real-world environment.

2. External force modeling experiment

To comprehensively evaluate the effectiveness of the two external force modeling methods based on learning the Jacobian matrix, multiple external force scenarios commonly encountered in actual manipulator interaction tasks are constructed in simulation. A thorough dynamics modeling and generalization experiment is conducted on the UR5 robot.

3. Modeling experiment under practical conditions

To assess the performance of the two models that consider the coupling between motor dynamics and external forces, DeLaN-MJF and DeLaN-MKF, in real-world interaction tasks, inverse dynamics modeling experiments are performed while the UR10e robot carries out material transport tasks. An incremental training strategy (10%–100% data) is applied to evaluate the model's learning efficiency.

4. Model-based control experiment

To evaluate the control performance of the models, model-based inverse dynamics control experiments are conducted in the simulation, comparing the trajectory tracking capabilities of different models.

4.1.2 Models

The abbreviations and full names of the different dynamic models are listed in Table 1.

The component-specific models DeLaN-motor, DeLaN-JF, and DeLaN-KF are compared against the baseline DeLaN model (Lutter et al., 2019) to validate the effectiveness of motor dynamics and the two external force modeling methods based on learning the Jacobian matrix, respectively.

A purely data-driven feedforward neural network (FFNN) (Lutter et al., 2019) is used as a baseline to validate the impact of embedding physical priors on the model performance and learning efficiency.

Three PINN-based models, namely the Lagrangian-based learned model (LNN) (Liu et al., 2024), the structured mechanical model of a control-affine system (SMM-C) (Gupta et al., 2020), and the deep Lagrangian network with feedforward neural network (DeLaN-FFNN) (Wu et al., 2024), are used as baselines to validate the model's ability to describe nonlinear features in complex interaction

Table 1 Abbreviations and full names of the different dynamic models

Model	Type	Full name of the model
FFNN	Baseline	Feedforward neural network
DeLaN		Deep Lagrangian network
DeLaN-FFNN		Deep Lagrangian network with feedforward neural network
LNN		Lagrangian-based learned model
SMM-C		Structured mechanical model of a control-affine system
DeLaN-motor	Proposed	Deep Lagrangian network with motor dynamics modeling
DeLaN-JF		Deep Lagrangian network with direct Jacobian force modeling
DeLaN-KF		Deep Lagrangian network with kinematic differentiation force modeling
DeLaN-MJF		Deep Lagrangian network with motor dynamics and direct Jacobian force modeling
DeLaN-MKF		Deep Lagrangian network with motor dynamics and kinematic differentiation force modeling

environments. LNN is a method based on PINNs that includes dissipation and allows non-isometric control actions. SMM-C introduces a black-box dissipative force network in the DeLaN architecture to learn non-conservative forces. DeLaN-FFNN integrates DeLaN with a standard deep network to model non-conservative dynamics (including nonlinear friction, external disturbances, and elastic effects) that are difficult to represent within the traditional Lagrangian framework.

4.1.3 Evaluation metrics

To comprehensively assess the performance of the proposed models, three metrics are employed: the normalized mean squared error (NMSE), the root mean squared error (RMSE), and the coefficient of determination (R^2). The R^2 metric is defined as

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\boldsymbol{y}_i - \hat{\boldsymbol{y}}_i)^2}{\sum_{i=1}^{n} (\boldsymbol{y}_i - \bar{\boldsymbol{y}})^2}, \tag{22}$$

where $\boldsymbol{y}_i$ and $\hat{\boldsymbol{y}}_i$ denote the actual and predicted motor currents (or joint torques), respectively, and $\bar{\boldsymbol{y}}$ represents the mean value of the dataset. To ensure balanced learning of the system's overall dynamics and prevent joints with larger torque magnitudes from dominating the prediction, equal weighting is applied to the R^2 score across all six joints.

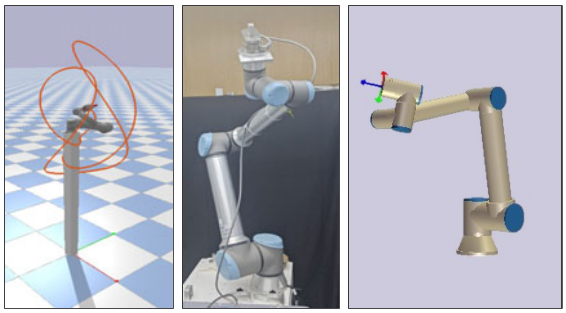
4.1.4 Dataset construction

1. UR5

The robot (Fig. 3a) is simulated using the PyBullet physics engine. To fully excite the system's dynamic characteristics while minimizing abrupt motions and impacts that could cause vibration or jerk, a sinusoidal-based reference trajectory is designed for the UR5 robot as follows:

$$\boldsymbol{q}_t^i = \boldsymbol{q}_0^i + \frac{\boldsymbol{q}_{\mathrm{f}}^i - \boldsymbol{q}_0^i}{T}\left(t - \frac{T}{2\pi}\sin\frac{2\pi t}{T}\right). \tag{23}$$

The trajectory ensures continuity and smoothness of the robotic motion by combining the initial position $\boldsymbol{q}_0^i$ and final position $\boldsymbol{q}_{\mathrm{f}}^i$ with a smooth transition function of time t and total duration T. A total of 200 distinct excitation trajectories are randomly generated, each lasting 8 s, with a sampling frequency of 50 Hz. The collected data include joint angles, velocities, torques, end-effector poses, end-effector velocities, and external forces for all six joints. Joint accelerations are computed using the central difference method.



(a) (b) (c)

Fig. 3 (a) UR5 robot in the simulation environment, demonstrating a trajectory of the end-effector from the test set; (b) real UR10e robot; (c) UR10e robot in the URsim simulation platform

2. UR10e

A communication link is established between the robot and the personal computer (PC) using the TCP/IP socket protocol to ensure command transmission to the robot and real-time data reception on the PC. Following the same procedure as for the UR5, 200 trajectories are randomly generated on the physical robot (Fig. 3b) and the URsim simulation platform (Fig. 3c), with a sampling frequency of 500 Hz. Joint angles, velocities, motor currents, end-effector poses, end-effector velocities, and external forces measured by the six-dimensional force/torque sensor at the end-effector are recorded. Joint accelerations are calculated using the central difference method.

Full-dimensional dynamic data from a typical material transport task are collected on the real robot. The transport task covers the following operating conditions: transient collision events during the transportation process (including 5–16 N impulse impacts in the $F_x/F_y/F_z$ directions), dynamically changing load masses (0–1.5 kg stepwise variation), continuous pushing and pulling transportation in the horizontal plane (3–11 N continuous forces in the F_x/F_y directions), vertical lifting operations (5–36 N varying loads in the F_z direction), and spatially varying loads during continuous multi-target point transportation (3–27 N combined forces in $F_x/F_y/F_z$ directions).

An ATI F/T Gamma six-dimensional force sensor is used to measure the external forces on the end-

effector in real time, with the following measurement resolutions in 6-DOF: 0.0125 N for F_x/F_y, 0.025 N for F_z, and 0.001 N·m for $M_x/M_y/M_z$ (M_x, M_y, and M_z represent the components of the applied torque around the x-, y-, and z-axis of the sensor, respectively). The sampling frequency is 500 Hz. To reduce the model training time, the collected data are downsampled to 50 Hz.

4.2 Inverse dynamics modeling experiment

4.2.1 Motor dynamics modeling experiment

First, modeling experiments are conducted on the UR10e robot in the ideal URsim simulation environment. All models are trained using the same dataset. Table 2 presents the inverse dynamics modeling results for the UR10e robot in this simulated environment, with corresponding visualizations provided in Fig. 4a. Bold values denote the optimal performance metrics across all evaluated models. A comparative analysis reveals that the DeLaN-motor model exhibits substantially reduced NMSE values for individual joints. It also shows a superior overall coefficient of determination ($R^2 = 0.9972$) compared to the FFNN baseline. This performance disparity arises from the FFNN's lack of physical constraints. In contrast, the DeLaN-motor architecture incorporates fundamental physical principles, resulting in enhanced predictive accuracy as evidenced by its near-unity R^2 value.

The results of the UR10e modeling experiments in real-world environments are presented in Table 2 and Fig. 4b. Both DeLaN-motor and FFNN demonstrate competent predictive performance with trends similar to those observed in the ideal simulation environment. The DeLaN-motor model achieves lower NMSE values for nearly all joints compared to FFNN. It also has an overall R^2 of 0.9857, confirming its ability to effectively learn the fundamental dynamics of the manipulator in physical deployments. These results collectively validate that the DeLaN-motor model, with integrated motor dynamics, exhibits superior modeling performance in both ideal simulation and real-world conditions. This demonstrates the efficacy of incorporating motor dynamics modeling.

Table 2 NMSE of motor currents and overall coefficient of determination (R^2) for each joint of the UR10e robot in both ideal simulation and real-world environments

Type	Model	NMSE						R^2
		Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	
Ideal simulation	FFNN	3.614e−1	2.008e−2	1.669e−2	7.832e−2	8.076e−2	1.689e−1	8.790e−1
	DeLaN-motor	**4.839e−3**	**2.077e−4**	**3.706e−4**	**6.979e−3**	**2.831e−3**	**1.280e−3**	**9.972e−1**
Real-world	FFNN	2.886e−2	2.485e−2	2.116e−2	1.335e−2	**1.605e−2**	2.509e−1	9.408e−1
	DeLaN-motor	**2.085e−2**	**1.068e−2**	**1.507e−2**	**1.176e−2**	1.708e−2	**1.029e−2**	**9.857e−1**

DeLaN: deep Lagrangian network; FFNN: feedforward neural network; NMSE: normalized mean squared error. Bold values indicate the better performance for each metric and joint

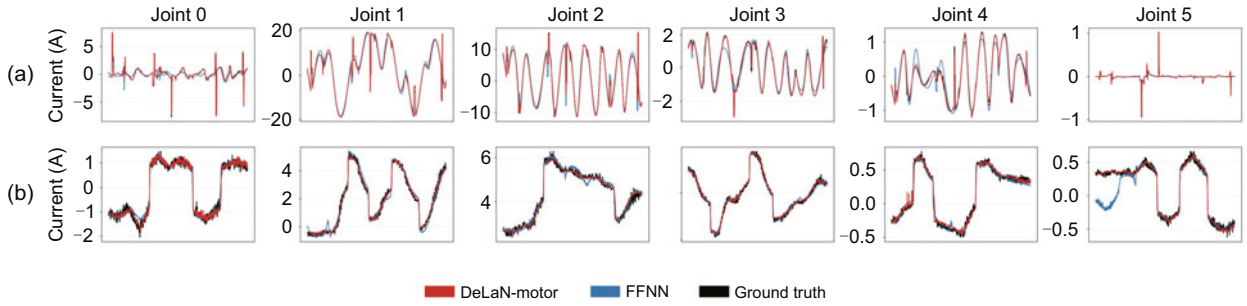


Fig. 4 Inverse dynamics learning performance of different models for the UR10e robot under zero external force: (a) modeling performance on the UR10e robot in an ideal simulation environment; (b) modeling performance on the real UR10e robot. The horizontal axis denotes the number of sample points. The black curve denotes ground truth, the red curve represents predictions of the DeLaN-motor model, and the blue curve corresponds to the FFNN model. DeLaN: deep Lagrangian network; FFNN: feedforward neural network. References to color refer to the online version of this figure

4.2.2 External force modeling experiment

1. Optimal parameter selection

In the PyBullet simulation platform, modeling experiments are conducted on the UR5 robot. First, the influence of the weight parameter λ on overall optimization is investigated to identify the optimal value. The DeLaN-KF model is tested under fixed external forces with a series of λ values. The experimental results are shown in Fig. 5. It is observed that when λ is small, the optimization objective is primarily dominated by dynamics. This results in lower accuracy of the learned Jacobian matrix and imprecise external force terms, which degrade overall model performance. As λ increases, kinematic optimization gains prominence, leading to a balance between system dynamics and kinematic optimization. Although some fluctuations occur, the overall performance shows an upward trend. However, when λ becomes excessively large, the optimization objective shifts to kinematic dominance. Although this results in precise Jacobian matrices, it adversely affects the learning of overall dynamics, thereby reducing modeling accuracy. Therefore, selecting an appropriate λ during model training is crucial for achieving optimal performance.

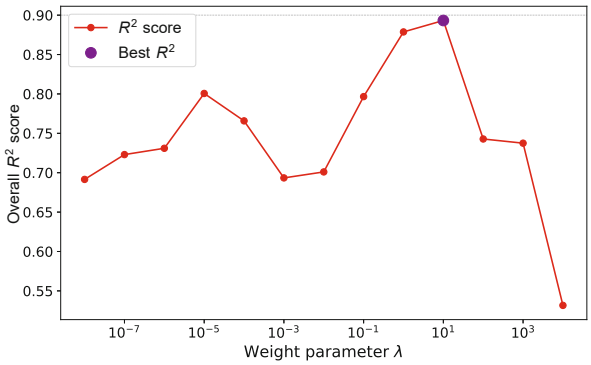


Fig. 5 Variation trend of the coefficient of determination (R^2) for the DeLaN-KF model with respect to the weight parameter λ. The purple data point indicates the λ value yielding the maximum overall R^2. References to color refer to the online version of this figure

2. Variable external force

Fig. 6 illustrates the performance evolution trends of different modeling methods in the UR5 robot system as a function of external force magnitude. Under smaller external forces, all models exhibit good learning capability, with R^2 values close to the ideal level. However, as the external force magnitude increases, the performance of the DeLaN and LNN models significantly deteriorates, especially when the external force exceeds 8 N, where their R^2 values fall outside the valid range of 0–1, indicating a complete loss of prediction reliability. Although the DeLaN-FFNN and SMM-C models show some improvement over the baseline DeLaN, their modeling accuracy remains limited, with noticeable performance degradation as the external force continues to increase. In contrast, the DeLaN-JF and DeLaN-KF models demonstrate excellent robustness, with their R^2 values remaining stable during the increase in external force without significant decline. Among them, the DeLaN-KF model consistently maintains the highest R^2 value across all test conditions.

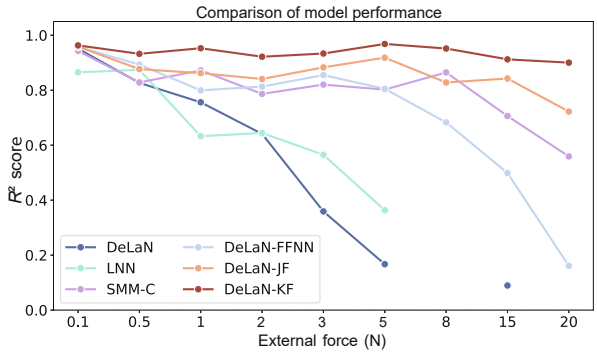


Fig. 6 Variation in the coefficient of determination (R^2) for each model's prediction of total joint torque across different magnitudes of applied external force, where omitted data points indicate R^2 values outside the [0,1] range. DeLaN: deep Lagrangian network; DeLaN-FFNN: deep Lagrangian network with feedforward neural network; LNN: Lagrangian-based learned model; SMM-C: structured mechanical model of a control-affine system

3. Multi-scenario force

To accurately simulate complex operating conditions of the real robot, this study systematically designs seven typical external force scenarios (as shown in Table 3). The modeling performance comparison results, as shown in Fig. 7, indicate that the DeLaN model maintains relatively high R^2 values only in two specific scenarios: high-frequency, low-amplitude vibrations and transient pulse impacts. However, in other operating conditions, the performance significantly deteriorates, with R^2 values generally below 0.6 or even exceeding the reasonable range of [0, 1]. The LNN model exhibits similar limitations

Table 3 Taxonomy of manipulator external force test scenarios

Scenario	Test case name	Force characteristic	Application context
1	Random disturbance	Randomly oriented forces	Collaborative robot dynamic contact
2	High-frequency vibration	Low-amplitude perturbations	Precision tasks under micro-vibration
3	Horizontal push/pull	Constant/varying horizontal forces	Assembly insertion, sliding motions
4	Vertical load/unload	Bidirectional vertical forces	Load lifting, press-fit operations
5	Cyclic force	Periodic forces	Spray painting, synchronized rotation
6	Sudden impact	Short-duration impulses	Emergency stops, accidental impacts
7	Gradual force increase	Quasi-static ramp force	Compliant assembly operation

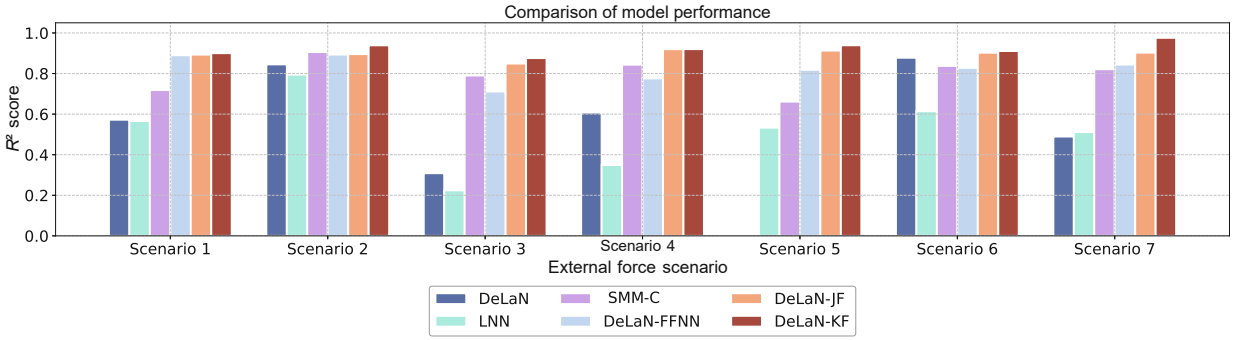


Fig. 7 R^2 performance of each model in predicting the resultant joint torque under different external force scenarios, where missing bars indicate R^2 values outside the $[0, 1]$ range. DeLaN: deep Lagrangian network; DeLaN-FFNN: deep Lagrangian network with feedforward neural network; LNN: Lagrangian-based learned model; SMM-C: structured mechanical model of a control-affine system

to the DeLaN model, with R^2 values generally low across the test scenarios, making it difficult to effectively capture the system dynamics under complex external forces. In contrast, the DeLaN-FFNN and SMM-C models show significant performance improvement, though their accuracy remains less than ideal in some scenarios. The DeLaN-JF and DeLaN-KF models consistently maintain excellent modeling performance across all test scenarios, with R^2 values stably above 0.85, confirming that the proposed methods exhibit superior adaptability to various operating conditions.

4. Generalization performance

To evaluate the generalization ability of the models under new external forces, all models are trained on an original small external force training set (1×) and tested on the same trajectories with the external force multiplied by a factor. The experimental results are shown in Fig. 8. Under the initial small external force condition, all models exhibit high modeling accuracy. As the external force magnitude increases, the performance of the models shows a differentiated degradation trend. Among them, the DeLaN model exhibits the most significant performance degradation, with R^2 dropping below 0.5 when the external force is increased 20 times. Although the LNN, DeLaN-FFNN, and

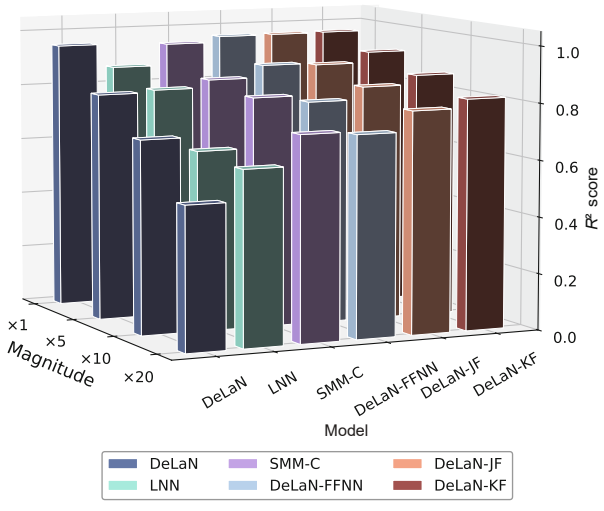


Fig. 8 Generalization performance evaluation: R^2 comparison across models under increasing external forces. DeLaN: deep Lagrangian network; DeLaN-FFNN: deep Lagrangian network with feedforward neural network; LNN: Lagrangian-based learned model; SMM-C: structured mechanical model of a control-affine system

SMM-C models experience relatively slow performance decline, noticeable degradation still occurs. In contrast, the DeLaN-JF and DeLaN-KF models demonstrate superior generalization, with the DeLaN-KF model maintaining the highest modeling accuracy across all test conditions. Even when the external force is increased 20 times, the R^2 value of the DeLaN-KF model remains above 0.8. This result verifies the strong adaptability of the proposed methods to new external force conditions.

The results of multiple external force modeling experiments indicate that the DeLaN model, due to the lack of an explicit non-conservative force modeling mechanism, has significant limitations in representing nonlinear external force characteristics, resulting in a considerable reduction in its modeling accuracy and generalization performance under external forces. In contrast, improved models such as LNN, SMM-C, and DeLaN-FFNN enhance the model performance to some extent by introducing neural network-based non-conservative force modeling methods. However, they still lack the ability to effectively represent complex nonlinear external forces.

Experimental data show that the DeLaN-JF and DeLaN-KF models, with their explicit external force modeling mechanisms achieved through Jacobian matrix fitting, demonstrate superior modeling performance and generalization ability in complex external force scenarios. In particular, the DeLaN-KF model, through the external force modeling method introduced via system kinematics, strictly follows the geometric constraints and physical laws of the manipulator. This not only ensures better physical consistency and interpretability but also exhibits optimal learning performance across all test scenarios, thoroughly validating the effectiveness of the proposed explicit external force modeling methods.

4.2.3 Modeling experiment under practical conditions

To evaluate the performance of the proposed models, DeLaN-MJF and DeLaN-MKF, in real-world robotic tasks, dynamic performance tests are conducted under a material transport scenario. The experimental results are shown in Table 4. The results indicate that the traditional purely data-driven black-box model, FFNN, exhibits good modeling performance but does not achieve ideal prediction accuracy. The PINN-based models, including LNN, SMM-C, and DeLaN-FFNN, do not show significant improvements in prediction accuracy compared to FFNN, and in some cases, even exhibit a certain degree of degradation. Among them, the LNN model performs particularly poorly. The analysis suggests that these models, by introducing black-box networks on top of DeLaN to fit unmodeled non-conservative force features, experience significant performance degradation in real-world task scenarios with prominent non-conservative forces due to insufficient fitting of the black-box components.

In contrast, the DeLaN-MJF and DeLaN-MKF models, which embed explicit prior modeling of friction and external forces, demonstrate superior prediction accuracy, especially on joints of the end-effector (Joint 5) that are significantly affected by external forces. The prediction error for these joints is notably lower than that of other comparative models. This advantage arises from the embedded structured

Table 4 NMSE of the current in the motors of each joint and overall coefficient of determination (R^2) of the UR10e under actual material transport conditions

Model	NMSE						R^2
	Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	
FFNN	9.660e−2	1.042e−1	3.603e−2	1.188e−1	8.947e−2	1.537e−1	9.002e−1
LNN	2.572e−1	2.127e−1	4.128e−1	1.768e−1	3.019e−1	2.770e−1	7.269e−1
SMM-C	4.397e−2	3.898e−1	7.540e−2	7.337e−2	3.922e−2	1.685e−1	8.683e−1
DeLaN-FFNN	1.084e−1	1.209−1	3.449e−2	5.199e−2	2.558e−2	1.557e−1	9.172e−1
DeLaN-MJF	2.398e−2	2.814e−2	4.810e−2	1.530e−2	2.294e−2	**1.045e−2**	9.752e−1
DeLaN-MKF	**2.212e−2**	**2.340e−2**	**2.315e−2**	**1.329e−2**	**1.746e−2**	1.132e−2	**9.815e−1**

DeLaN-FFNN: deep Lagrangian network with feedforward neural network; DeLaN-MJF: DeLaN with motor dynamics and direct Jacobian force modeling; DeLaN-MKF: DeLaN with motor dynamics and kinematic differentiation force modeling; FFNN: feedforward neural network; LNN: Lagrangian-based learned model; NMSE: normalized mean squared error; SMM-C: structured mechanical model of a control-affine system. Bold values indicate the best performance for each metric and joint

physical knowledge in the model architecture, ensuring that the predicted results are consistent with physical laws.

It is noteworthy that the DeLaN-MKF model demonstrates remarkably superior predictive performance across nearly all tested joints, indicating that enhancing physical interpretability and consistency through learning system kinematics can further improve modeling performance. However, for certain rigid robotic systems with complex structures, such as highly redundant manipulators or those with complex parallel architectures, the kinematic relationships are often highly nonlinear and difficult to accurately describe using conventional analytical methods like Denavit–Hartenberg (DH) parameters. The kinematic differentiation approach used in DeLaN-MKF is often inadequate for effectively learning the complex kinematic models of such systems. In comparison, although DeLaN-MJF exhibits slightly lower performance than DeLaN-MKF, it treats the velocity mapping between joint and task space as a locally linearized problem. This method directly learns the differential relationship in a data-driven manner without requiring an explicit kinematic model, thereby showing stronger applicability in scenarios involving complex system structures or modeling challenges. Additionally, DeLaN-MJF requires end-effector velocity data for training, which in many applications can be acquired more directly and cost-effectively. This ease of data acquisition further enhances the practical adaptability of DeLaN-MJF in real-world systems.

All inference time tests are performed on the same computer equipped with an NVIDIA RTX 3060 GPU (6 GB). The purely data-driven black-box model FFNN obtains the fastest inference time of 6.754 78e−4 s due to its simple network structure. DeLaN-MJF directly learns the mapping from joint velocities to end-effector velocities and outputs the Jacobian matrix through a single forward propagation, with an inference time of 8.361 98e−4 s. Although slightly slower than FFNN, it performs better than other compared models. The baseline models LNN, SMM-C, and DeLaN-FFNN show similar inference speeds, with times of 9.383 96e−4 s, 1.025 81e−3 s, and 8.928 28e−4 s, respectively, all slightly slower than DeLaN-MJF. In contrast, DeLaN-MKF requires additional learning of the system kinematic model and analytical differentiation, leading to a more complex computational process, and therefore achieves the longest inference time of 1.206 93e−3 s. Although significantly slower than DeLaN-MJF, its inference speed still meets the real-time requirements of most industrial manipulator applications.

To evaluate the efficiency of dynamic learning in the model and verify the advantages of modeling based on the PINN approach with limited data, the model is trained 10 times using 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, and 100% of the training dataset, as shown in Fig. 9. When the training data reaches 90%, the purely data-driven model FFNN achieves stable accuracy. Without the guidance of physical priors, the black-box model requires a large amount of data to learn the complex nonlinear mapping relationship of the system and is highly dependent on the data. The PINN-based models (LNN, SMM-C, and DeLaN-FFNN) achieve stable performance with 50%–70% of the training data, demonstrating better learning efficiency than the purely data-driven method. The proposed models, DeLaN-MJF and DeLaN-MKF, by embedding friction and external force models as physical priors into the network, achieve stable performance with only 40%–50% of the training data. More importantly, both models under the condition of low data volume exhibit high prediction capabilities even with limited data.

4.3 Model-based control experiments

In this experiment, to systematically evaluate the control performance of different models, we employ the trained neural network as a feedforward dynamical model within the computed torque control framework. The control law is formulated as follows:

$$\boldsymbol{\tau} = \boldsymbol{H}(\boldsymbol{q})\boldsymbol{u} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}), \tag{24}$$

where $\boldsymbol{u}$ is the reference acceleration input, composed of the desired acceleration and a proportional-derivative (PD) compensation term:

$$\boldsymbol{u} = \ddot{\boldsymbol{q}}_{\mathrm{d}} + K_{\mathrm{p}}(\boldsymbol{q}_{\mathrm{d}} - \boldsymbol{q}) + K_{\mathrm{v}}(\dot{\boldsymbol{q}}_{\mathrm{d}} - \dot{\boldsymbol{q}}), \tag{25}$$

where $\boldsymbol{q}_{\mathrm{d}}$, $\dot{\boldsymbol{q}}_{\mathrm{d}}$, and $\ddot{\boldsymbol{q}}_{\mathrm{d}}$ represent the desired joint angle, velocity, and acceleration of the manipulator, respectively. K_{p} and K_{v} are the feedback gains. To highlight the role of dynamic model prediction in control, the feedback gains K_{p} and K_{v} are set to

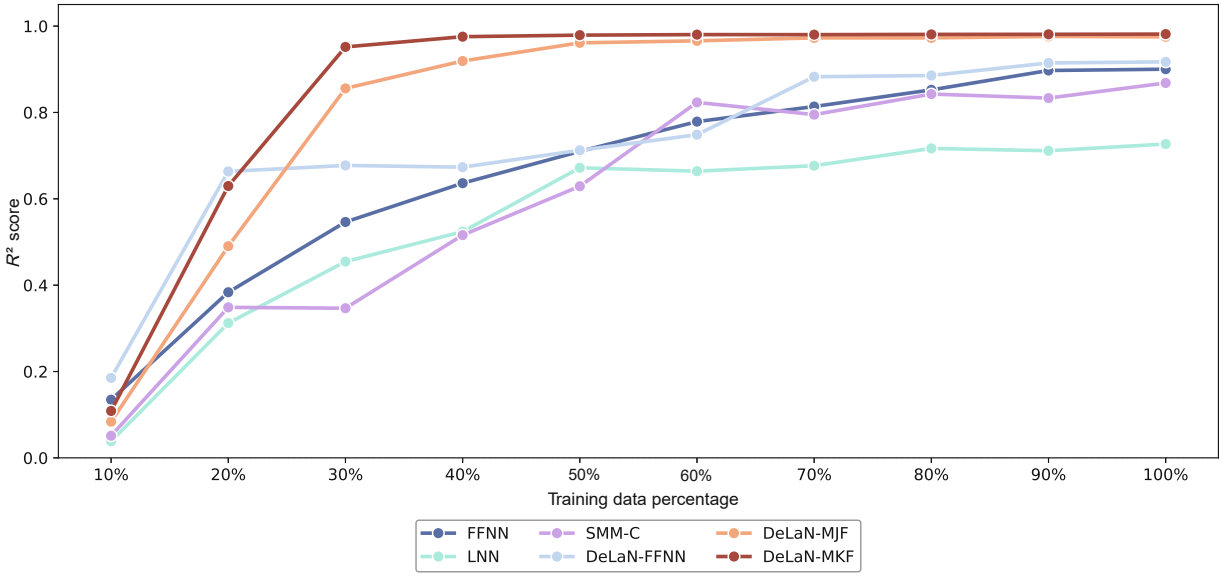


Fig. 9 Comparison of modeling accuracy R^2 of different models under varying dataset sizes. DeLaN-FFNN: deep Lagrangian network with feedforward neural network; DeLaN-MJF: DeLaN with motor dynamics and direct Jacobian force modeling; DeLaN-MKF: DeLaN with motor dynamics and kinematic differentiation force modeling; FFNN: feedforward neural network; LNN: Lagrangian-based learned model; SMM-C: structured mechanical model of a control-affine system

relatively low values in the experiments, making the tracking error more sensitive to model accuracy.

Using the same desired trajectory as the training data, real-time trajectory tracking tests under external disturbances are conducted on a 6-DOF UR5 manipulator in the PyBullet simulation environment. Fig. 10 presents the tracking performance of different models over one cycle, while Table 5 details the tracking errors for each joint. The results demonstrate that compared to the baseline DeLaN-based controller without non-conservative force modeling, the controllers incorporating DeLaN-FFNN, DeLaN-JF, and DeLaN-KF models achieve significantly higher tracking accuracy. This improvement is particularly pronounced at Joint 5, which exhibits strong nonlinear characteristics, where the DeLaN controller shows a substantially higher RMSE of 1.576. Although the DeLaN-FFNN controller demonstrates notable performance enhancement over the baseline DeLaN controller—achieving the lowest RMSE at Joint 4 among all controllers—its overall tracking precision remains limited. Both the DeLaN-JF and DeLaN-KF controllers demonstrate favorable tracking performance, with the RMSE of DeLaN-JF being comparable to that of DeLaN-FFNN. Notably, the DeLaN-KF controller achieves the lowest RMSE in nearly every joint, with particularly superior performance at Joint 5, demonstrating the

Table 5 RMSE of trajectory tracking for different control models across the UR5 robot joints

Control model	RMSE					
	Joint 0	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5
DeLaN	0.242	0.753	0.838	0.794	0.282	1.576
DeLaN-FFNN	0.100	0.431	0.377	0.550	**0.096**	1.046
DeLaN-JF	0.058	0.216	0.259	0.580	0.102	1.374
DeLaN-KF	**0.039**	**0.208**	**0.157**	**0.532**	0.143	**0.664**

DeLaN: deep Lagrangian network; DeLaN-JF: deep Lagrangian network with direct Jacobian force modeling; DeLaN-FFNN: deep Lagrangian network with feedforward neural network; DeLaN-KF: deep Lagrangian network with kinematic differentiation force modeling; RMSE: root mean squared error. Bold values indicate the best performance for each joint

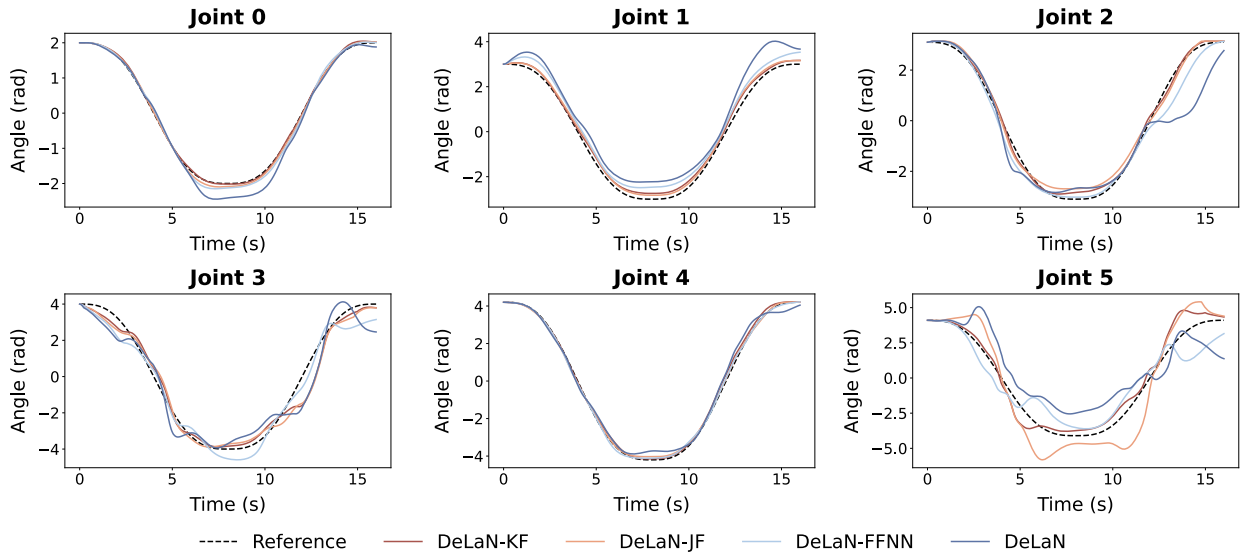


Fig. 10 Trajectory tracking performance of different models on all six joints of the UR5 robot. DeLaN: deep Lagrangian network; DeLaN-FFNN: deep Lagrangian network with feedforward neural network; DeLaN-KF: deep Lagrangian network with kinematic differentiation force modeling; DeLaN-JF: deep Lagrangian network with direct Jacobian force modeling

most robust control capabilities overall.

It is observed that the tracking error of Joint 5 is significantly higher than that of other joints. This occurs because, in the simulation environment, all external forces are set to directly act on the end-effector (Joint 5). To better compare the control performance between models, a model with significant external forces is used, which significantly affects the dynamics of Joint 5 and reduces the accuracy of its dynamic modeling. Taking the optimally performing DeLaN-KF model as an example, its NMSE is 3.08 times higher than the average of the other joints. This insufficient dynamic modeling accuracy lowers the accuracy of feedforward compensation in torque control, directly affecting control performance. Additionally, the error accumulation effect along the kinematic chain propagates to the end-effector, resulting in a cumulative effect with the modeling error of Joint 5, which collectively affects the final control accuracy. The poor control performance of Joint 5 can lead to risks such as decreased manipulator positioning accuracy, delayed dynamic response, and force control instability. This is used only to compare the control performance between different models. In practical tasks, more precise control strategies are required to meet the high-precision control demands of the end-effector.

5 Conclusions

This paper proposes two PINN frameworks, DeLaN-MKF and DeLaN-MJF, which integrate motor dynamics and external force modeling. Both frameworks incorporate explicit external force modeling through two distinct Jacobian matrix estimation methods, along with motor dynamics priors, effectively addressing the challenge of modeling in the absence of joint torque sensors. Experiments conducted on simulated and physical manipulators demonstrate that the proposed models achieve outstanding modeling accuracy, generalization capability, and control performance across various dynamic interaction scenarios. In particular, DeLaN-MKF achieves optimal performance in most scenarios by implicitly embedding geometric constraints through learning system kinematics, highlighting the positive impact of physical interpretability on performance improvement. Although slightly inferior in absolute accuracy, DeLaN-MJF directly learns the velocity mapping from the joint space to task space without complex differentiation processes, resulting in higher computational efficiency and making it more suitable for systems with higher real-time requirements. Moreover, it exhibits stronger adaptability to robotic systems with complex kinematic structures, such as parallel mechanisms and redundant manipulators.

Future work will focus on developing optimization strategies for hybrid models and intelligent control methods to further enhance the trajectory tracking accuracy and dynamic response performance of manipulators, as well as to validate the generalizability of the approach across a wider range of robot configurations and complex manipulation tasks.

Contributors

Fengyu SUN and Shuangshuang WU designed the research, conducted the experiments, and analyzed the results. Zhiming LI and Peilin XIONG assisted with data processing. Fengyu SUN drafted the paper. Shuangshuang WU and Wenbai CHEN reviewed and revised the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Akbari M, Mehr JK, Ma L, et al., 2023. Uncertainty-aware safe adaptable motion planning of lower-limb exoskeletons using random forest regression. *Mechatronics*, 95:103060. https://doi.org/10.1016/j.mechatronics.2023.103060

Caccavale F, Siciliano B, Villani L, 2003. The Tricept robot: dynamics and impedance control. *IEEE/ASME Trans Mechatron*, 8(2):263-268. https://doi.org/10.1109/TMECH.2003.812839

Chen ZX, Renda F, Le Gall A, et al., 2025. Data-driven methods applied to soft robot modeling and control: a review. *IEEE Trans Autom Sci Eng*, 22:2241-2256. https://doi.org/10.1109/TASE.2024.3377291

Cheng S, Hu BB, Wei HL, et al., 2025. Deep learning-based hybrid dynamic modeling and improved handling stability assessment for autonomous vehicles at driving limits. *IEEE Trans Veh Technol*, 74(4):5582-5593. https://doi.org/10.1109/TVT.2024.3515209

Chrosniak JY, Ning J, Behl M, 2024. Deep dynamics: vehicle dynamics modeling with a physics-constrained neural network for autonomous racing. *IEEE Robot Autom Lett*, 9(6):5292-5297. https://doi.org/10.1109/LRA.2024.3388847

Clochiatti E, Scalera L, Boscariol P, et al., 2024. Electromechanical modeling and identification of the UR5 e-series robot. *Robotica*, 42(7):2430-2452. https://doi.org/10.1017/S0263574724000833

Crane III CD, Duffy J, 1998. Kinematic Analysis of Robot Manipulators. Cambridge University Press, New York, USA.

Djeumou F, Neary C, Goubault E, et al., 2022. Neural networks with physics-informed architectures and constraints for dynamical systems modeling. Proc 4th Learning for Dynamics and Control Conf.

Duong T, Altawaitan A, Stanley J, et al., 2024. Port-Hamiltonian neural ODE networks on Lie groups for robot dynamics learning and control. *IEEE Trans Robot*, 40:3695-3715. https://doi.org/10.1109/TRO.2024.3428433

Gu WB, Primatesta S, Rizzo A, 2024. Physics-informed neural network for quadrotor dynamical modeling. *Rob Auton Syst*, 171:104569. https://doi.org/10.1016/j.robot.2023.104569

Gupta JK, Menda K, Manchester Z, et al., 2020. Structured mechanical models for robot learning and control. Proc 2nd Annual Conf on Learning for Dynamics and Control, p.328-337.

Hu HB, Shen ZK, Zhuang CG, 2025. A PINN-based friction-inclusive dynamics modeling method for industrial robots. *IEEE Trans Ind Electron*, 72(5):5136-5144. https://doi.org/10.1109/TIE.2024.3476977

Khatib O, 1987. A unified approach for motion and force control of robot manipulators: the operational space formulation. *IEEE J Robot Automat*, 3(1):43-53. https://doi.org/10.1109/JRA.1987.1087068

Laddach K, Łangowski R, Rutkowski TA, et al., 2022. An automatic selection of optimal recurrent neural network architecture for processes dynamics modelling purposes. *Appl Soft Comput*, 116:108375. https://doi.org/10.1016/j.asoc.2021.108375

Lahoud M, Marchello G, D'Imperio M, et al., 2024. A deep learning framework for non-symmetrical Coulomb friction identification of robotic manipulators. Proc IEEE Int Conf on Robotics and Automation, p.10510-10516. https://doi.org/10.1109/ICRA57147.2024.10610737

Li ZL, Bai JS, Ouyang HJ, et al., 2024. Physics-informed neural networks for friction-involved nonsmooth dynamics problems. *Nonl Dyn*, 112(9):7159-7183. https://doi.org/10.1007/s11071-024-09350-z

Li ZM, Wu SS, Chen WB, et al., 2024. Extrapolation of physics-inspired deep networks in learning robot inverse dynamics. *Mathematics*, 12(16):2527. https://doi.org/10.3390/math12162527

Li ZM, Wu SS, Chen WB, et al., 2025. Physics-informed neural networks for compliant robotic manipulators dynamic modeling. *J Comput Sci*, 90:102633. https://doi.org/10.1016/J.JOCS.2025.102633

Liu JY, Borja P, Della Santina C, 2024. Physics-informed neural networks to model and control robots: a theoretical and experimental investigation. *Adv Intell Syst*, 6(5):2300385. https://doi.org/10.1002/aisy.202300385

Lutter M, Peters J, 2023. Combining physics and deep learning to learn continuous-time dynamics models. *Int J Rob Res*, 42(3):83-107. https://doi.org/10.1177/02783649231169492

Lutter M, Ritter C, Peters J, 2019. Deep Lagrangian networks: using physics as model prior for deep learning. Proc 7th Int Conf on Learning Representations.

Phong LD, Choi J, Lee W, et al., 2015. A novel method for estimating external force: simulation study with a 4-DOF robot manipulator. *Int J Precis Eng Manuf*, 16(4):755-766. https://doi.org/10.1007/s12541-015-0100-7

Pikuliński M, Malczyk P, Aarts R, 2025. Data-driven inverse dynamics modeling using neural-networks and regression-based techniques. *Multibody Syst Dyn*, 63(3):341-366. https://doi.org/10.1007/s11044-024-10024-2

Roehrl MA, Runkler TA, Brandtstetter V, et al., 2020. Modeling system dynamics with physics-informed neural networks based on Lagrangian mechanics. *IFAC-PapersOnLine*, 53(2):9195-9200. https://doi.org/10.1016/j.ifacol.2020.12.2182

Shah SV, Saha SK, Dutt JK, 2012. Modular framework for dynamic modeling and analyses of legged robots. *Mech Mach Theory*, 49:234-255. https://doi.org/10.1016/j.mechmachtheory.2011.10.006

Sousa CD, Cortesão R, 2014. Physical feasibility of robot base inertial parameter identification: a linear matrix inequality approach. *Int J Robot Res*, 33(6):931-944. https://doi.org/10.1177/0278364913514870

Tao Y, Chen S, Liu HT, et al., 2025. Robot hybrid inverse dynamics model compensation method based on the BLL residual prediction algorithm. *Robotica*, 43(1):350-367. https://doi.org/10.1017/S0263574724001905

Trinh M, Geist AR, Monnet J, et al., 2025. Newtonian and Lagrangian neural networks: a comparison towards efficient inverse dynamics identification. https://arxiv.org/abs/2506.17994

Wu SS, Li ZM, Chen WB, et al., 2024. Dynamic modeling of robotic manipulator via an augmented deep Lagrangian network. *Tsinghua Sci Technol*, 29(5):1604-1614. https://doi.org/10.26599/TST.2024.9010011

Xu PF, Han CB, Cheng HX, et al., 2022. A physics-informed neural network for the prediction of unmanned surface vehicle dynamics. *J Marine Sci Eng*, 10(2):148. https://doi.org/10.3390/jmse10020148