Applied Soft Computing Journal 163 (2024) 111877

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Physics informed machine learning model for inverse dynamics in robotic manipulators

Weikun Deng [a,*], Fabio Ardiani [b], Khanh T.P. Nguyen [a], Mourad Benoussaad [a], Kamal Medjaher [a]

[a] Laboratoire Génie de Production, LGP, Université de Toulouse, INP-ENIT, 47 Av. d'Azereix, 65016, Tarbes, France
[b] Nimble One, 8 rue du Canard, 31000, Toulouse, France

ARTICLE INFO

Keywords:
Physics-informed machine learning
Robotic manipulators
Parameter identification
Equation embedded neural network
Liquid mechanism

ABSTRACT

In the field of robotic modelling, the challenge of parameter estimation using limited joint monitoring data presents a substantial hurdle for both traditional physics-based methods (PBMs) and machine learning (ML) techniques. PBMs grapple with modelling uncertainties, variable working conditions, diverse robotic configurations, and incomplete parameter information. ML methods face hurdles in maintaining physical consistency, interpretability, and the need for extensive training data. In response to these challenges, this paper proposes a novel approach, the Equation Embedded Neural Network (E2NN), enhanced by an innovative Liquid mechanism, which effectively blends the strengths of PBMs and ML to surmount their inherent limitations. Its primary contributions encompass: (1) a pioneering review and synthesis of hybrid frameworks integrating physical principles with ML, (2) the development and rigorous validation of the E2NN framework, and (3) the introduction of a novel physics regulator and dynamic Liquid mechanism. Particularly, the proposed E2NN leverages inverse dynamics equations to construct specialized neural network layers, featuring activation functions and interconnections expressed as composition operators, thus explicitly encoding physical knowledge. This architectural choice proves especially well-suited for tasks involving inverse dynamics and dynamic planning of robotic manipulators. The accompanying Liquid mechanism allows for dynamic adaptation of interlayer connections in response to input data, enabling real-time adjustments to changing inputs and equations of motion, ultimately enhancing flexibility and performance. Quantitative assessments of E2NN reveal its compelling performance, yielding a Mean Absolute Error (MAE) of 0.10716, closely aligned with the Benchmark Deep Residual Shrinkage Network's (DRSN) MAE of 0.10415, showcasing its competitive efficacy while achieving higher computational efficiency and a more compact model size. Robustness evaluations further confirm E2NN's adaptability, as it attains a Mean Squared Error (MSE) of 0.3, outperforming DRSN's 1.1 under varying working conditions. E2NN excels in torque trajectory fitting, achieving an impressive accuracy rate of 97.1%, underscoring its practical effectiveness. Furthermore, E2NN excels in torque prediction and parameter identification for inverse dynamics models, particularly when confronted with limited joint data and variable friction conditions. It substantially improving the discernment of robot dynamics and enhancing its applicability in real-world trajectory fitting.

1. Introduction

In the field of robotics, the best approach for control is the use of model-based techniques [1]. The more precise the model is, the better and faster the performance of the controller will be. The foundation of these models lies in the intricate mathematical interplay of physical quantities governed by Newton's laws which consider the relate states, as position, velocity, acceleration and forces to a big number of parameters including masses, inertia tensors, distances between bodies, and more [2].

Unluckily, obtaining precise real numerical values for the aforementioned parameters is not a straightforward task, and numerous robot manufacturers opt to conceal this information for logical safety and copyright reasons. An alternative for measuring these values involves analysing and conducting experiments on individual components of the robot separately [3]. However, this requires the risky and time-consuming disassembly of the robot, with the added complication that the assembly process itself may introduce significant changes. Another viable approach is to use a Computer-Aided Design (CAD) model of the robot. Nevertheless, such models may be incomplete and may not

* Corresponding author.
E-mail address: weikun.deng@enit.fr (W. Deng).

https://doi.org/10.1016/j.asoc.2024.111877
Received 25 June 2023; Received in revised form 22 May 2024; Accepted 10 June 2024
Available online 14 June 2024

account for certain factors, such as friction, flexibility and parameter variations due to external factors as temperature and forces.

The third approach, and mostly used in robotics, is to carry out a parameter identification process (also called gray-box modelling). It obviates the necessity to disassemble the robot and has proven to be more accurate than the other methods. It relies on the statistical matching between real sampled data from the robot and the mathematical formulation of the model [4]. It consists of an iterative process, involving multiple stages, each with its distinct challenges and design options, as which physical phenomena should be included on the model and how, which trajectory should be chosen, and which mathematical technique should be used to obtain the result, many of which were addressed in [5]. Some comprehensive surveys of parameter identification in robotics can be found in [5,6].

Several statistical methods have been proposed and tested in literature based on the classical dynamic equation of the robots [7]. Some of the solutions are based on the well-known Least-Squares technique [8,9], which has been proven to be the best linear unbiased estimator if the error terms have zero mean. They are homoscedastic (constant finite variance), do not present multi-collinearity and are not autocorrelated between them. As these assumptions are most of the times difficult to ensure, other techniques have been used, such as those based on the Instrumental Variable [10,11], the so-called Output Error methods [12,13] and the Input Error methods [14]. Furthermore, for control techniques as adaptive control [2], the tracking of the parameters on real-time is important. For this reason, the recursive variants of these methods have been proposed [5,15]. Additionally, the mathematical approach may produce physically implausible results, such as negative mass values, because of noise and certain assumptions. Therefore, alternative methods have been suggested to ensure physical validity [16,17], including the utilization of the widely recognized Extended Kalman Filter algorithm [18]. These mathematical methods mentioned possess various merits and drawbacks concerning computational cost, speed, complexity, noise robustness, sensitivity to initial conditions, resistance to perturbations and modelling inaccuracies, real-time suitability, physical coherence, as well as accuracy and precision. Nonetheless, they share a common reliance on the representation of the robot's dynamic model. Most of them exploit the fact that the model (especially the inverse dynamic model) can be written in the linear form of $y = Ax + \rho$, where y are the torques needed to generate a specific movement, x are the set of parameters to be identified, ρ is the noise, perturbations and not modelled effects, and A is called the observation matrix, which includes the measurements of position, velocity and acceleration of the different parts.

This raises several challenges. First, the increasing complexity of robots makes it harder to symbolically derive observation matrices. Second, incorporating non-linear effects such as flexibility [19], friction [20], and environmental factors like temperature and load [21] limits the applicability of many existing methods. Third, the temporary insignificance of certain parameters due to sub-excitation [22], and the influence of disturbances and unaccounted factors require adaptive methods. Specifically, in robotic arm joints, constructing a friction model like LuGre [23,24] with dynamic parameters is often essential to accurately estimate friction under varying working conditions, in conjunction with the standard dynamics model.

In this context, Machine Learning (ML)-based data-driven methods, such as neural networks (NNs), offer alternative approaches from a physics-agnostic perspective (also included in the so called black-box modelling methods). The main advantages lie in the adaptability to cope with the robotic model variations in real-time [25]. ML methods focus on fitting the relations and time dependencies between system parameters and motion data by stacking multiple ML models [26], allowing for the capture of complex relations and accurate predictions without the need for detailed mechanism analysis or physics-based induction [27]. Although these methods seem attractive, it is crucial to acknowledge their inherent limitations. First, the training process necessitates a substantial quantity of labelled data and computational resources. Second, over-fitting the network in the training data will make it fail to generalize effectively to unseen data. Thirdly and most importantly, the absence of physics consistency and interpretability within the generated models, make it complicated to use the results for other purposes (i.e. control and identification) than simulation.

Recognizing the inherent limitations in both traditional statistical methods and ML-based approaches, the focus has shifted towards developing hybrid methods that combine the strengths of both, aligned with the concept of dark-grey box modelling introduced by Ljung in [28]. Therefore, our paper commences by thoroughly reviewing advanced hybrid methods in Section 2, pinpointing their shortcomings. This comprehensive analysis constitutes our **first significant contribution**. Building upon the principles of Physical informed Machine Learning (PIML) as delineated in [29], we then develop the Equation Embedded Neural Network (E2NN). This novel paradigm, detailed in Section 3, is our **second and major contribution**. The framework captures the understanding of robotic arm dynamics within a physics operator embedded network structure. This network features inter-layer connections that adapt to input data, allowing it to account for changes dynamically and autonomously in knowledge across various robotic arm states. **Our third contribution** is the empirical validation of E2NN's efficacy. Detailed in Section 4, this validation process employs both simulation data and real-world scenarios across varying conditions, utilizing a novel metric to measure the performance. This step underscores the practical applicability and effectiveness of the E2NN framework in real-world settings. Finally, in Section 5, we conclude with a summary and outline perspectives for future research.

2. Related works

A hybrid framework in the context of robotic dynamics is commonly understood as the development of a semi-parametric model, representing a confluence of gray-box (partially physics-based) and black-box (data-driven) methodologies. This study delineates a novel threefold classification of state-of-the-art (SOTA) hybrid frameworks, categorizing them based on the specific functions of ML within these systems. The classification encompasses three distinct roles:

1. Physics assisted system surrogate (PAS): ML signifies an unknown term in a physical equation, thereby contributing to the final system model in a hybrid ML-equation form.
2. Physics augmented estimation (PAE): ML estimates system parameters that facilitate the resolution of kinetic equations and validate the adherence to analytical laws.
3. Physics assisted matching (PAM): ML identifies suitable dynamic solutions by matching system models and parameters across the entire workspace, within a known selection scope.

Within that perspective, Table 1 summarizes the related researches.

These investigations consistently highlight a myriad of advantages, encompassing improved performance, augmented data efficiency, and increased physical consistency. Notably, emphasis has been placed on the integration of dynamical equations into NN structures [31], and the construction of NNs subject to physical constraints [32]. Particularly, in the field of soft-bodied robots [34], the amalgamation of traditional incomplete analytical models with machine learning (ML) has showcased positive effects. Nonetheless, the envisioned future trajectory of these works necessitates a more in-depth exploration of model flexibility, a comprehensive examination of the methodologies employed to embed physics knowledge into diverse ML structures, and practical validation.

The Physics assisted system surrogate (PAS) methodology has found successful applications in various fields, including the Universal Robots for 5 kg weights (UR5) manipulator [30] and soft pneumatic actuators [34]. These models offer advantages such as enhanced data efficiency and adaptability. However, they encounter challenges when

Table 1
Summary of hybrid methods for robotic manipulator identification. (Ta-Taxonomy).

Application	Hybrid method	Advantages	Future work	Ta.
UR5 manipulators.	Combines imitation learning and reinforcement learning (RL). ML is initialized by solving a supervised learning problem that uses the expert's state and action pairs to learn the inverse dynamics [30].	Without requiring demonstrator action information, learning directly from a single demonstration trajectory.	Learn a generalized controller from numerous demonstrations.	PAS
Barrett WAM, Kuka-LWR robot.	Lagrangian induction semi-parametric for estimating robotic parameters, and equation consistency [31,32].	Guarantee physically plausible dynamics.	Only simulate articulated rigid bodies without contact and relies on knowing and observing the generalized coordinates.	PAS
A benchmark suite of robotics environments.	Physics-based side information shapes network structure and constraints output values and internal states [33].	Improves data efficiency and generalization of the model.	Use the physics-informed dynamics models to design control laws.	PAS
Soft pneumatic actuators.	Embedding physical models into RNN recursive computations [34].	Enhanced prediction accuracy and effectiveness across diverse types of RNNs and soft robotics platforms.	Apply PIRNN to more practical engineering problems in soft robotics.	PAS
MICO and Fetch robotic manipulator.	Using Generative Adversarial Networks to compensate analytical models' approximation errors [35].	More data efficiency, better performance, and accuracy.	Reduce the redundancy and computational cost. Try to adapt to sparse data.	PAS
Da Vinci surgical robot.	Design each joint's NN that receives input measurements from every robot joint and outputs torque/force estimation, using the identification error for training supervision [36].	The identification method can be improved with more data sets and training.	The necessary force information may not always be available in a surgical setting.	PAE
Dielectric elastomer actuators control.	Differentiable model combining a material properties neural network for PBM parameter estimations and an analytical dynamics model for responses calculation [37].	$\lesssim$5% simulation error compared to finite element models and low time cost.	Improve control and inference algorithms for real and multi-DOF soft robot.	PAE
Multi-link robotic manipulators.	The derivatives of the residuals at a finite number of matched points were designed as physical loss functions to guide the PINN to approximate the true responses [38].	Efficient gradient-based algorithms for the underlying optimal control problem.	Enhance PINNs to handle more complex control actions and initial values.	PAE
KUKA manipulators.	Use meta-learning to learn structured, state-dependent loss functions for updating the model parameters [39].	Quick adaptation to changes in dynamics.	Investigate the state-dependent loss. Explore its performance in more parameters' estimation.	PAE
Simulated 7-dof manipulator.	Two novel physics-informed loss functions about the space velocity control error [40,41].	Superior performance to using normal deep models and easy to design.	Extending to multi-body contact, explore the effect of adaptively tuning the trade-off loss weight.	PAE
Simulated multi-DOF manipulators.	Optimizing end-effector position with wave function and Monte Carlo method to satisfy the Euclid fitness through the particle swmanipulators optimization [42,43] or artificial bee colony [44].	Low computation cost and acceptable performance on the simulation data.	Testing on various robotic manipulators, exploring different fitness functions, and evaluating its practical performance.	PAM
Different multi-joint robotic manipulators, such as ABB Industrial Robot.	Artificial Neural Networks [45], Multi-layer perceptron, Long Short-Term Memory, Gated Recurrent Unit [46], Fuzzy NN [47] nonlinearly fits and matches end-effector pose to joint configurations in the entire working space.	Easy to design, high precision and model the uncertain factors.	Explore the model's flexibility and the identification of key influencing parameters in the model, and expand its application.	PAM
ICub humanoid robot inverse dynamics modelling [48]	Semiparametric models that combine rigid body dynamics (parametric model) with Gaussian processes (nonparametric model). Two versions proposed: with RBD in the mean function or with RBD in the kernel.	Better generalization, derivative-free	Compare with parametric method, enhance physical, design for predictive Control	PAM

it comes to generalization from limited demonstrations and computational efficiency. One significant limitation is their inability to adapt to diverse scenarios and different operating conditions. This limitation arises because, when machine learning is employed to approximate unknown terms within physical equations, all connections of the approximation operators and data flow become fixed, hard to learn and modify.

On the other hand, PAE methods have been implemented in systems like the DaVinci surgical robot [36] and multi-link robotic manipulators [45]. These methods excel in estimating system parameters, boasting high simulation accuracy and efficient control algorithms. However, they often struggle to maintain physical consistency, particularly in the presence of non-model effects like friction and temperature variations. Ensuring the realism and physical accuracy of these models across various conditions remains a significant challenge.

PAM strategies, which involve machine learning to match dynamic solutions across a wide range of workspaces, can be observed in applications such as the ICub humanoid robot [48]. While proficient at matching dynamic solutions in diverse situations, they grapple with the complexity and computational demands associated with handling intricate robotic systems. These challenges encompass processing and managing the computational load of advanced robotic systems in real-time, while also requiring improved physical consistency and comparative analyses with parametric approaches.

Furthermore, these studies primarily focus on improving control performances through ML-based nonlinear data fitting, rather than finding fixed patterns and estimating the system parameters. While hybrid PBM-ML methods are advancing, there is a significant gap in achieving an optimal balance between physical consistency and real-time adaptability under the ML framework where physical models can be expressed. This gap highlights the need for more sensitive inputs and knowledge flexible architectural approaches in hybrid modelling. Consequently, the primary objective of this paper is to introduce a versatile and interpretable PIML framework tailored for robotic manipulator modelling and parameter identification. This framework consists of a physics-informed structure method with the ability to generalize on new data from unknown working conditions. Hereby, in the validation process, this study will demonstrate the effectiveness of the proposed design by comparing the performance before and after using the proposed framework to modify an existing algorithm.

3. Proposed framework

Motivated by the research published in [49] which proposes implementing the flux–tendency relations directly into establishing NN structure to inherently respect the conservation laws, this research develops the penultimate hidden layer and interlayer connections enforcing robot dynamics in the proposed E2NN. In addition, for adapting to the different working modes, a liquid inter-layer connection paradigm is explored. The proposed framework is shown in Fig. 1.

The implementation process involves building an inverse dynamics model using analytic derivation first, to study the individual terms of position, velocity and acceleration that are repeated over and over. Next, the NN's neurons are designed based on basic computational functions in the analytical relations incomplete model, such as sine and cosine. These neurons form layers that express sub-term transformation relations for the inverse dynamics model. Then inter-layer connections are designed based on the relations between sub-terms. Finally, a fully connected layer represents the summation operation relations. Each step of the computation is implemented through substitution using the NN. A detailed description of the proposed framework is shown in the following sections, especially the innovative parts are given in Section 3.2.

3.1. Robot inverse dynamic model

The inverse dynamics model of a m degrees of freedom robotic manipulator can be written, as explained in Section 1, in a linear way with respect to its parameter as:

$$\boldsymbol{\tau} = W(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})\beta + \rho \tag{1}$$

where $W(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) \in \mathbb{R}^{r \times n_b}$ is the observation matrix in Eq. (2) obtained from evaluating $f_i(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}})$ m times, with $r = n \times m$, and n the amount of measurements; $\rho \in \mathbb{R}^r$ are the unmodelled effects; $\tau \in \mathbb{R}^r$ is the torque vector of the robotic manipulator; $\mathbf{q}$, $\dot{\mathbf{q}}$, and $\ddot{\mathbf{q}}$ are joint positions, velocities, and accelerations measurements, respectively; and $\beta \in \mathbb{R}^{\mathrm{n}_b}$ is the vector of the n_b base parameters describing the manipulator.

$$W(\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}) = \begin{bmatrix} f_1(q_1, \dot{q}_1, \ddot{q}_1) \\ f_2(q_2, \dot{q}_2, \ddot{q}_2) \\ \vdots \\ f_m(q_m, \dot{q}_m), \ddot{q}_m \end{bmatrix} \tag{2}$$

The parameter set β is selected to minimize model complexity while ensuring that there is no multicollinearity within the rows of the observation matrix. This particular parameter set is referred to as the base parameter used in the field of robotics. It constitutes a linear transformation of the conventional dynamic parameters, which include the centres of mass, inertial properties, and masses of the robotic links, given that the kinematic parameters (the relative distances and angles between joints) are presumed to be known [50,51]. The derivation of this base parameter set can be approached through analytical methods [50], as well as through numerical strategies [52,53]. The premise of treating the kinematic parameters as known is a prevalent assumption in robotics. This is due to the ease of accurately measuring link dimensions with contemporary measurement technologies. In instances where these kinematic parameters are unavailable or where increased precision is required, kinematic calibration techniques can be employed to refine these values [54].

As also said in Section 1, the trajectory selection is an important task on parameter identification. It will determine if parameters are well-excited or not, thus it will determine the confidence on the estimation of them [55,56]. For manipulators that are commanded through position, velocity and/or acceleration, the well-known finite Fourier series is the most popular way to design the reference trajectory to excite as many parameters as possible [57,58]:

$$\begin{aligned} \mathbf{q_i}(t) &= \sum_{l=1}^{N_i} \left(\frac{a_{l,i}}{\omega_f l} \sin(\omega_f l t) - \frac{b_{l,i}}{\omega_f l} \cos(\omega_f l t) \right) + q_{i0} \\ \dot{\mathbf{q}}_\mathbf{i}(t) &= \sum_{l=1}^{N_i} \left(a_l \cos(\omega_f l t) + b_l \sin(\omega_f l t) \right) \\ \ddot{\mathbf{q}}_\mathbf{i}(t) &= \sum_{l=1}^{N_i} \left(-a_l \omega_f l \sin(\omega_f l t) + b_l \omega_f l \cos(\omega_f l t) \right) \end{aligned} \tag{3}$$

where ω_f represents the fundamental pulsation of the Fourier series, and N_i indicates the order of the harmonic. The parameters a_l and b_l correspond to the amplitudes of the sine and cosine functions, while q_0 denotes the initial position around which the joint oscillates.

Moreover, friction needs to be added to the basic dynamic model. Many models exist in literature [20,59,60]. One of the most used in robotics is:

$$\tau_{fj} = f_{vj}\dot{q}_j + f_{cj}\mathrm{sign}(\dot{q}_j) + \tau_{offj} \tag{4}$$

where τ_{fj} represents the joint friction torque; f_{vj} is the viscous friction coefficient; f_{cj} is the Coulomb friction coefficient; and τ_{offj} is used to combine the asymmetrical Coulomb friction coefficient and other offsets caused by sensors and amplifiers [61].

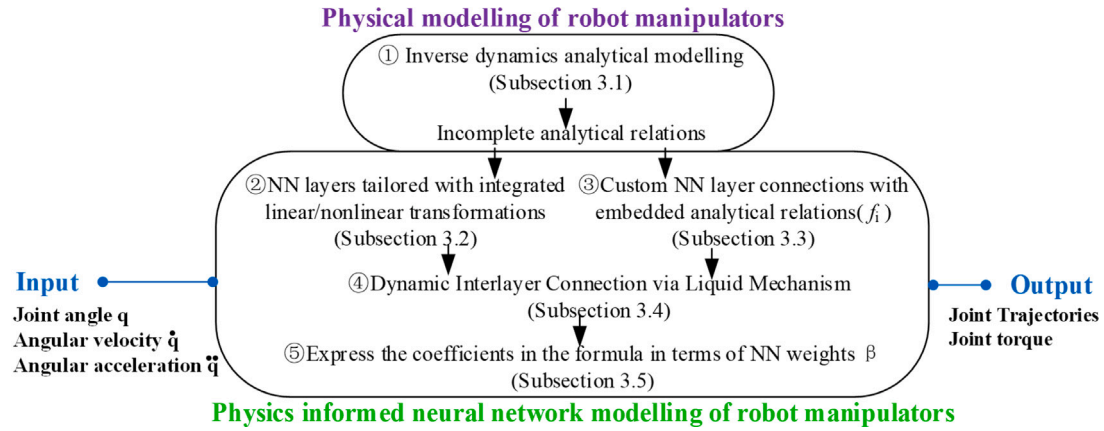


Fig. 1. Implementation framework for E2NN.

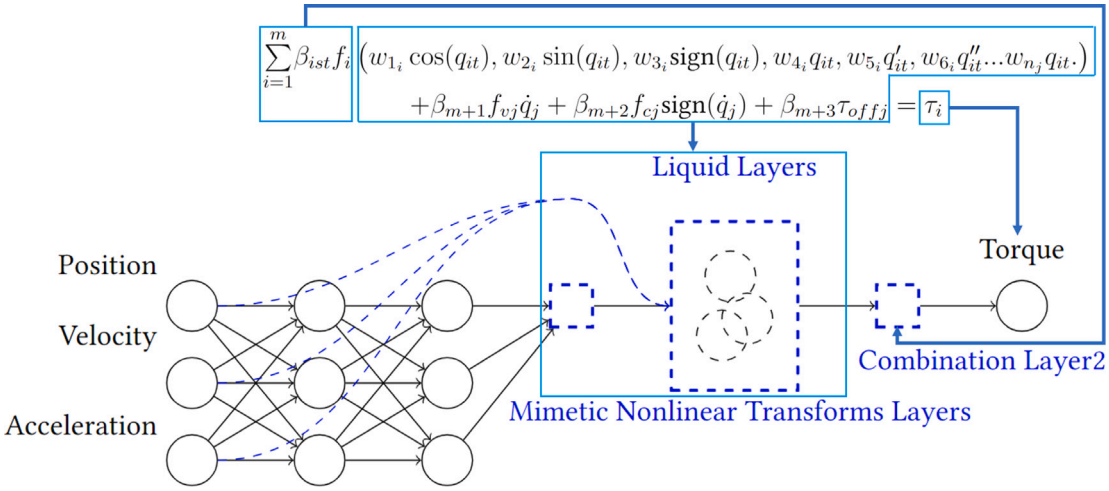


Fig. 2. Architecture of the Equation-Embedded Neural Network (E2NN) for one robotic joint. (The blue components represent the physics-informed parts of the traditional neural network. The circular nodes represent individual neurons, while the rectangular nodes denote custom layers. The interconnected circular group within the liquid layer signifies their dynamic connections. In the Liquid layers, the connection is indeterminate and is therefore represented by the stacking pattern of the neuron pool. The relationship between the physical equation and the NN structure is given by the dark blue arrow.).

By careful study of the dynamic equation, for each joint, we can rewrite Eq. (1) in order to build informed-NN as:

$$\tau_i = \sum_{p=1}^{P}\beta_p f_{ip}\left(\mathbf{w}_{1_{\mathbf{pi}}}\cos(\mathbf{q}), \mathbf{w}_{2_{\mathbf{pi}}}\sin(\mathbf{q}), \mathbf{w}_{3_{\mathbf{pi}}}\text{sign}(\mathbf{q}), \mathbf{w}_{4_{\mathbf{pi}}}\mathbf{q}, \mathbf{w}_{5_{\mathbf{pi}}}\dot{\mathbf{q}}, \mathbf{w}_{6_{\mathbf{pi}}}\ddot{\mathbf{q}}...\right) + \beta_{P+1}\dot{q}_i + \beta_{P+2}\text{sign}(\dot{q}_i) + \beta_{P+3} \tag{5}$$

where P are the amount of parameters related to τ_i. The detailed mathematical expressions for the overall nonlinear mapping function f_{ip} and the coefficients β and w are to be learned from monitoring data. This model is, of course, a simplification of reality, and that is why we say in this paper that it is incomplete. Several phenomena may be missing and/or modelled differently.

3.2. Proposed methodology

The process of ML modelling is to learn parameters β_p. These parameters are combined with different transformations. Additionally, the learned relations need to be flexible to accommodate variations in working conditions and adapt to different input data. To satisfy these requirements, a specific NN structure is designed according to Eq. (5). The details are shown in Fig. 2 and Section 3.2.1. The architecture's core is underpinned by the implementation of custom design. Within the "Mimetic Nonlinear Transforms Layers", the foundational physical relations are simulated for each input to effectively create a symbolic representation of the physics involved. Subsequent to this transformation, the "Combination Layers" amalgamate this symbolic relation into an output that aligns with the prescribed dynamic formulation. In the "Combination Layers", the weights, symbolizing the parameters of the robotic manipulators, are subject to automated adjustment through an iterative learning process.

3.2.1. NN layers with embedded domain equations

E2NN uses the customer layers to simulate the basic nonlinear transformation terms, as shown in Fig. 4 (see Fig. 3).

Where h_i represents a series of partially known physics nonlinear elements f in Eq. (5). n_b is the type number of the basic items involved. h_i consists of the basics mathematical terms such as $\cos(q)$, $\sin(q)$, $\text{sign}(\dot{q})$, and linear transformation relations of q, $\dot{q}$, $\ddot{q}$. For the unknown elements in Eq. (5), E2NN uses the traditional ML layer h_u to fit the potential transformation. Thus far, E2NN has performed the polynomial fitting of the independent variables contained in f_i for Eq. (5). In particular, the bias term b of the last hidden layer corresponds to τ_{offj}.

In the deeper hidden layers, E2NN designs custom layers to simulate the basic mathematical combinations of the above basic elements, as illustrated in Fig. 4.

Where k_i represents a series of known mathematical relations consisting of squares, radicals, various types of quadratic operations, etc.

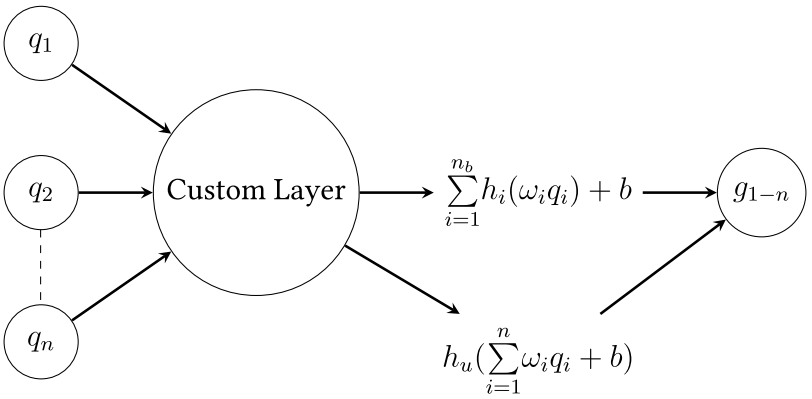


Fig. 3. Custom layers to represent basic arithmetic units.

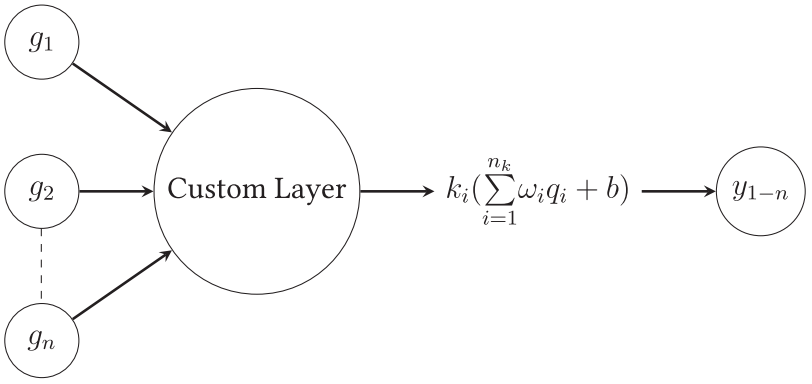


Fig. 4. Custom layers to represent analytic combination relations for polynomials.

n_k is the type number of the basic combination relations involved in Eq. (5).

3.2.2. NN interlayer connection

To fit a polynomial function of degree n_j to a set of data points in Eq. (5), $n+1$ coefficients are needed to define the polynomial. These pending coefficients β_i, where i ranges from 1 to n_j, can be represented by the weights and biases of a single hidden layer with n_j neurons. The input of these hidden neurons is responsible for providing the values of f in Eq. (5), and the output neuron produces the final predicted torque value τ.

3.2.3. Liquid mechanism to enhance the model generalization performance

The structure presented in Fig. 5 incorporates the dynamics equation as a hard constraint, resulting in a model with strong physical consistency. However, the existing PIML works have fixed node and connectivity configurations after training. This restricts the model's potential for adapting to novel working conditions, particularly in cases where the underlying equations differ across diverse operating conditions. When it comes to a straightforward prediction task, training a complex neural network for a mobile manipulator robot is not an optimal choice in terms of time efficiency [62] so that the connectivity between nodes of the proposed model in E2NN must be able to change dynamically. Therefore, the liquid mechanism is proposed to build custom dynamics inter-layer connection based on the current input and previous state of the network. Inspired by the "Liquid time-constant neural network " research published in [63], this paper introduces the "Liquid mechanism" to address the robust performance issue. A gating unit has been incorporated between the physics-informed layers. It comprises a gating controller and a nonlinear transformer which regulate the information flow and transformation between the layers, as shown in Fig. 6. The colours represent the different connectivity patterns when the E2NN layer processes different inputs. These connectivity patterns refer to the number of connections, between each neuron and other neurons, and the Eq. (6) based gating control connection weights.

$$\begin{aligned} z &= \sigma(W_z h + U_z l) \\ g &= \tanh(W_g h + U_g(\alpha \odot h)) \\ k' &= (1-z) \odot k + z \odot g \end{aligned} \tag{6}$$

In the formula (6), h is the input vector of the current layer, k is the output vector of the current layer, z is the gating unit, g is the nonlinear transformer, W_z, U_z, W_g, U_g are model parameters, σ, and $\tanh$ are activation functions, and $\odot$ denotes element-wise multiplication. The third equation in the formula indicates that the new output k' is calculated by a non-linear transformation g controlled by the gating unit z, in combination with the previous layer output.

It is noted that the leakage rate α plays a key role in controlling the scale of the output h of the upper level in the nonlinear transformation g. Typically, α is related to the time constant τ, to which is assigned a pre-determined value. However, to achieve input-related dynamics leakage rate and consider the dynamic physics constraints inside, α can be calculated using $\alpha = sigmoid(h)$.

3.2.4. Interpretable parameters

The parameter β of Eq. (5) in the E2NN represents the weights of polynomial terms that correspond to channel weights in the hidden layer of a neural network, as shown in Fig. 5. To enhance the performance of the E2NN, a customized loss function that takes into account both the layer parameters and the underlying physics equations has been developed and incorporated into the loss calculation. This ensures that the E2NN can achieve optimal values for both the NN structural parameters and the polynomial weights during training. In Fig. 7. The loss function is composed of two parts: the mean squared

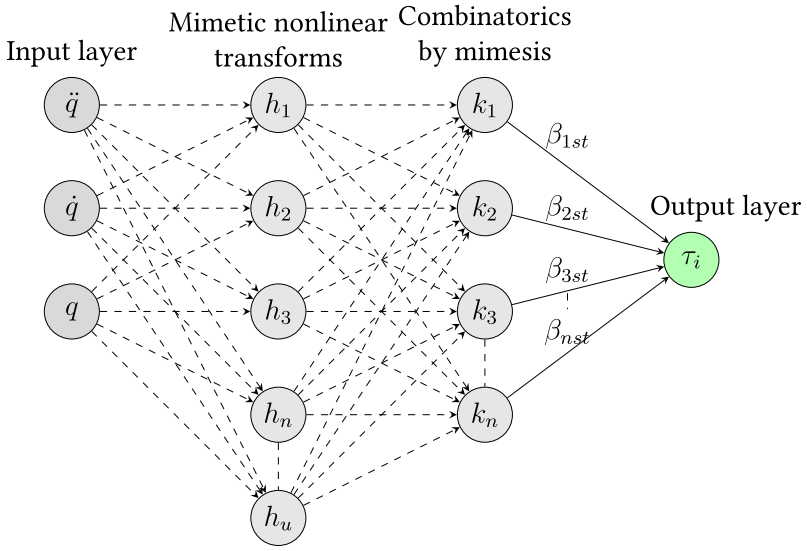


Fig. 5. The structure of the custom inter-layer connection in the E2NN layers (the dashed line in the figure indicates that the hidden layer in the middle is omitted).

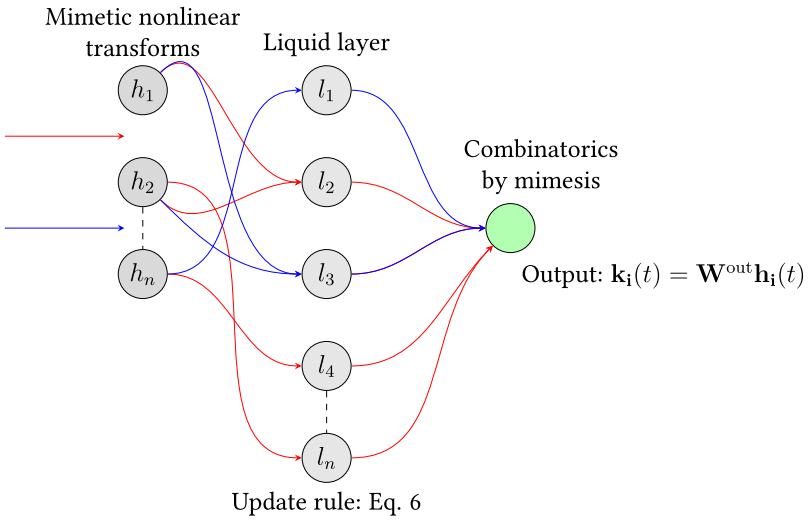


Fig. 6. E2NN layer with variant input dependent connections and hidden state update rule.

error (MSE) between the predicted (τ') and true (τ) values, and a physical formula term that accounts for the dynamics of the robotic manipulators movements. The physical formula term is defined as:

$$loss = \frac{1}{n_k}\sum_{i=1}^{n_k}(\tau - \tau')^2 + \frac{1}{n_k}\sum_{i=1}^{n_k}(\tau - \sum_{i=1}^{n_j}\beta_{inn} f_i\left(q_{it}, \dot{q}_{it}, \ddot{q}_{it}\right))^2 \tag{7}$$

4. Verification of the proposed methodology in limited data conditions

The efficacy of the E2NN is affirmed through simulation and experimental procedures conducted on a robotic manipulator, as will be detailed in Section 4.1. The architectural choice for this paper is the Residual Block-E2NN, which employs a residual neural network as a critical component of the E2NN framework that is introduced in Section 4.2. The evaluation metrics employed to assess the performance of the proposed methods are explained in Section 4.3. Validation results are shown and analysed in Sections 4.4 and 4.4.2.

4.1. Application cases

For our experimentation, we utilized the 7-degree-of-freedom (DOF) collaborative manipulator KUKA iiwa, as depicted in Fig. 8, along with its Denavit–Hartenberg parameters. This manipulator is characterized by a baseline of 43 base parameters, which extends to 64 when incorporating the 21 parameters related to the friction model as described by Eq. (4) in research [15]. In an effort to streamline our study and focus on a subsystem, we narrowed our analysis to only the last link of the robot. The dynamics of this link are described in Eq. (4), see τ given in Box I.

where $q, \dot{q}, \ddot{q}, \tau$ refer to values of the 7th joint, and the numerical values arise for combination of the kinematic parameters, which are considered to be provided and known. Notice the fundamental mathematical operations on the measurement signals mentioned in Section 3.2.1.

In this study, we made a strategic decision to focus exclusively on the last link of the 7-DOF KUKA iiwa manipulator, primarily to simplify the inherent complexity of a full robotic arm system. This targeted approach enables us to delve deeply into the dynamics and control

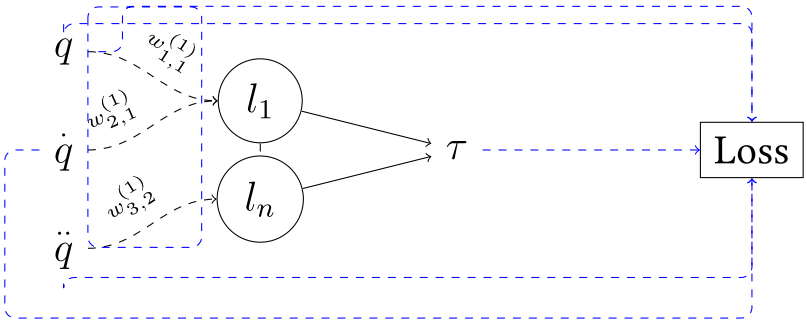
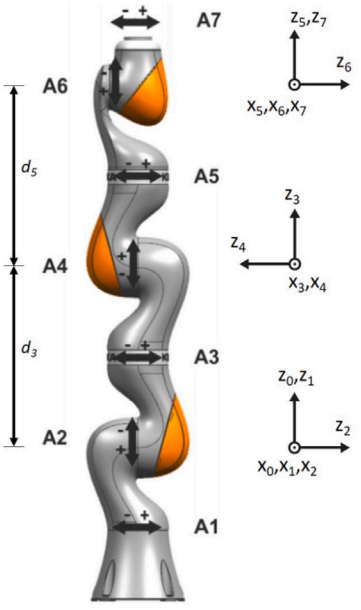


Fig. 7. Dynamics equation embedded loss function. The blue dashed line indicates the parameter used to calculate the final loss, and the black dashed line represents the intermediate hidden layer that passes from input q to output τ.

$$\tau = \begin{bmatrix} 0.06\cos(q) - 0.22\sin(q) \\ (0.22\cos(q) + 0.06\sin(q)) \\ -0.04\ddot{q} \\ (0.06\cos(q) - 0.22\sin(q))^2 - (0.22\cos(q) + 0.06\sin(q))^2 \\ \dot{q}(0.22\cos(q) + 0.06\sin(q)) - 0.006\sin(q) - (\dot{q} + 0.49)(0.22\cos(q) + 0.06\sin(q)) - 0.1098\cos(q) \\ 0.11\sin(q) - 0.0062\cos(q) - (\dot{q} + 0.49)(0.06\cos(q) - 0.22\sin(q)) + \dot{q}(0.06\cos(q) - 0.22\sin(q)) \\ 4.90\sin(q) - 0.026\cos(q) \\ 4.90\cos(q) + 0.026\sin(q) \\ sign(\dot{q}) \\ \dot{q} \\ 1.0 \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{10} \end{bmatrix} \tag{8}$$

Box I.



Link i	θ_i	d_i	a_i	α_i
1	q_1	0	0	0
2	q_2	0	0	$-\pi/2$
3	q_3	$0.42m$	0	$\pi/2$
4	q_4	0	0	$\pi/2$
5	q_5	$0.40m$	0	$-\pi/2$
6	q_6	0	0	$-\pi/2$
7	q_7	0	0	$\pi/2$

Fig. 8. Parameters and link frames of the KUKA LBR iiwa 14 R820 [9].

algorithms of a specific, yet representative, segment of the manipulator. However, it inherently limits the breadth of our findings. Specifically, the results may not fully encapsulate the dynamic interactions among the multiple links of a full robotic manipulator because the embedded operators in Eq. (4) are limited. Therefore, when extrapolating to more complex, multi-link manipulator systems, the extend of the NN structure and scale are required.

Regarding the kinematic parameters of the robotic system, we operated under the assumption that these parameters are known and accurately provided. This assumption is grounded in the fact that

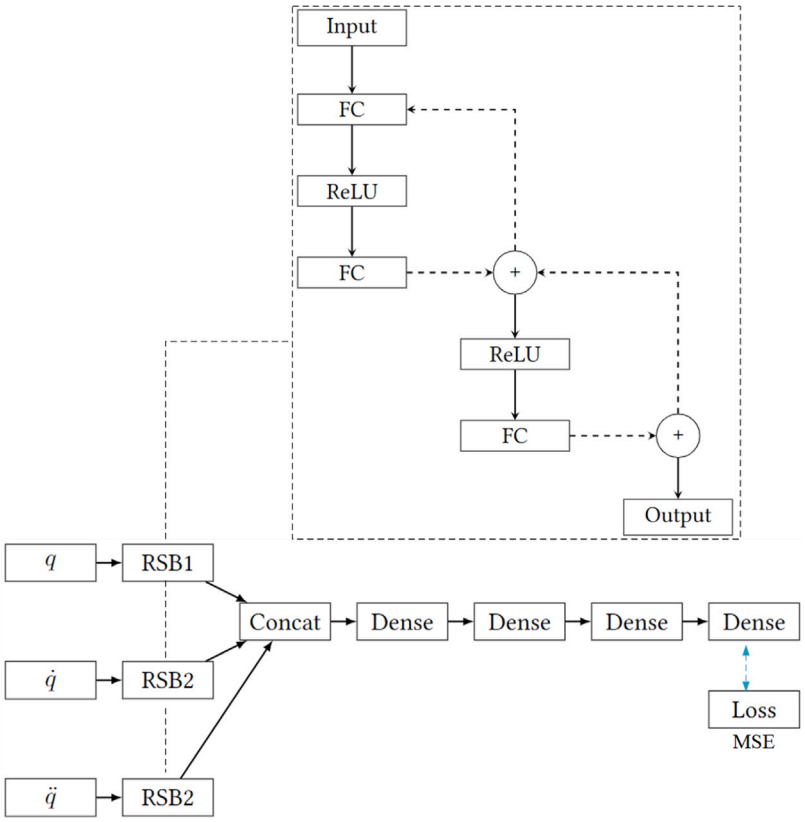


Fig. 9. The framework of the deep residual shrinkage network (DRSN).

robotic systems, like the KUKA iiwa, usually have well-defined kinematic parameters, either from precise manufacturing specifications or through calibration processes. However, this assumption may not fully capture the challenges encountered in real-world scenarios, where manufacturing variances and operational wear-and-tear can lead to deviations in these parameters. Consequently, while this assumption facilitated a more streamlined analysis, it potentially overlooks the nuances of parameter identification and adjustment in practical settings. The performance and accuracy of our model, therefore, might need adjustments or recalibrations when applied in operational environments where these parameters could differ from their initially defined values. But the resulting bias can be attributed to a robustness problem, which is expected to be compensated by the ML part of E2NN.

4.1.1. Simulation test

To generate the simulation environment of the manipulator, the following steps are considered:

1. Define the robot's dynamic equations.
2. Specify the robot's physical parameters, obtained from previous identification processes as the one in [15].
3. Design the trajectories to be used. In this work several trajectories were used, some of them that excite predominantly the phenomena of friction (called "Friction"), others that excite the mainly the inertial parameters (called "Inertial"), and others that excite both (called "Direct-Servo") with trajectories designed as mentioned in Eq. (3).
4. Use the provided equations and inputs to emulate the robot's motion over a specified duration using numerical integration methods like the Euler or Runge–Kutta.

The training data consist of 26,988 measurements made on several different trajectories, mainly exciting inertia parameters. The testing data include 53,976 movement recording points from trajectories mainly exciting friction parameters, to analyse the generalization ability of the proposed approach. For testing robustness, additional data is used as a second training dataset, but the E2NN avoids retraining on it.

4.1.2. Experimental test

In the experimental validation, the KUKA Sunrise OS is used to interpolate the trajectory points and create the displacement, velocity, and acceleration profiles with the Spline and PTP motion types [15]. The Fast Robot Interface (FRI) [64] library provided by KUKA is used to continuously exchange data in real time between the robot controller and a C++ client application on an external system. The client application recorded data from the robot at its highest possible rate of 1000 Hz. For the commanded signals, the data are first down-sampled to 50 Hz, and then velocities and accelerations are calculated from the commanded position. A second-order digital Butterworth filter with a cutoff frequency of 3.5 Hz in both directions is applied before the down-sampling process. A total of 42977 robot manipulator pose data points from 10 random task trajectories were collected, of which 3 tasks totalling 11,894 data points were randomly selected as the training set, and the remaining 7 tasks totalling 31,103 data points were used as the test set to simulate real small-sample conditions.

4.2. E2NN with residual blocks

The E2NN framework enhances a deep residual shrinkage network (DRSN) architecture, as illustrated in Fig. 10, forming the E2NN-ResNet model. This is contrasted with a conventional ResNet model, depicted in Fig. 9. Our comparative analysis centres on the structural differences between standard artificial neural networks (ANNs) based on residual blocks and an advanced version integrating a Physics-Informed (PI) module. The conventional ResNet lacks this PI module, whereas the E2NN-ResNet incorporates it, notably adding links between the PI module and the corresponding PI layer. This addition in E2NN-ResNet introduces a unique residual block employing trigonometric

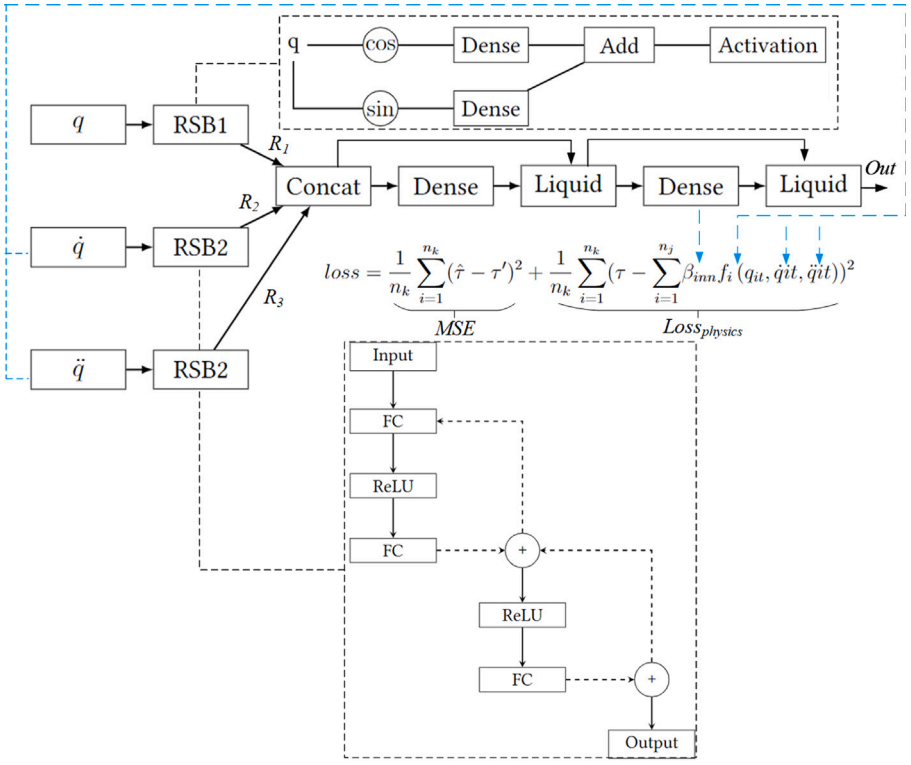


Fig. 10. E2NN: Deep Residual Shrinkage Network with an embedded equation proposal.

operations (sine and cosine functions) to compute sums and differences of inputs, a functionality absents in the standard ResNet structure. This structural enhancement in E2NN-ResNet is pivotal for understanding the distinctions and improvements brought about by incorporating PI principles in ANN architectures. In this paper, deep residual shrinkage network (DRSN) has three inputs, all of which yield a (batch size, 1) input for the subsequent layers. It consists of 6 residual blocks with a fully connected layer. The output of the block is summed with a shortcut connection by adding a residual connection with the ReLU activation function. This model uses the mean square error of torque prediction as the loss function and is trained using the Adam optimizer.

The E2NN is innovatively implemented to refine the DRSN model by modifying the activation function and the interconnections within the residual block, thereby highlighting the distinctiveness of E2NN in Fig. 10. The E2NN demonstrates considerable prowess in mimicking the inverse dynamics process subjected to physical constraints, thus offering a notable improvement to existing methods. The structure of the model comprises Residual Shrinkage Blocks and a Liquid Layer. The first of the Residual Shrinkage Blocks, "RDB1", calculates the cosine of q (shortcut = tf.cos (q)) and maps it to the filters via a dense layer. The second block "RDB2" processes $\dot{q}$ and $\ddot{q}$. The "Concat" layer then merges the three outputs processed by the residual shrinkage block. Additionally, their products ($R1 \times R1$, $R1 \times R2$) are concatenated. The "Liquid" layer assimilates these components and fits them to τ in an approximate mathematical form. This methodical and systematic approach to equation representation and assimilation forms the essence of the E2NN's superior functionality.

4.3. Evaluation metrics

To quantify the performance of the proposed method regarding the adaptation degree of the robot manipulators trajectory, mean square error (MSE), the Fréchet distance [65], and the Polygon Area difference [66]) are used as metric.

4.3.1. Polygon area difference

Polygon area is calculated by using Shoelace's formula Eq. (9):

$$\text{Area} = 0.5 \cdot \left| \sum_{i=0}^{n-2} (q_i \cdot y_{i+1} - q_{i+1} \cdot y_i) + (q_{n-1} \cdot y_0 - q_0 \cdot y_{n-1}) \right| \tag{9}$$

where (q_i, y_i), $i = 0, 1, \ldots, n-1$, characterize the polygon vertices. Next, the absolute difference between the areas of the predicted and observed trajectories is shown in Eq. (10).

$$\text{D} = \left| \text{Area}_{\text{predicted}} - \text{Area}_{\text{actual}} \right| \tag{10}$$

A smaller value of D indicates a better match between the predicted and the actual trajectories.

4.3.2. Fréchet distance

This metric measures the similarity between two curves, taking into account the location and arrangement of points. It is particularly useful for comparing robot manipulators' trajectories as it takes into account their spatial and temporal aspects. Given two curves P and Q represented by sequences of points, the Discrete Fréchet Distance (DFD) can be calculated by dynamic programming. The equation for the DFD is recursively defined as follows:

$$\text{DFD}(P, Q) = c(|P|, |Q|) \tag{11}$$

Where $c(i, j)$ is a function defined as:

$$c(i,j) = \begin{cases} d(P_i, Q_j) & \text{if } i = 1 \text{ and } j = 1 \\ \max\left(d(P_i, Q_j), \right. & \left. \min(c(i-1, j), c(i-1, j-1), c(i, j-1))\right) \\ & \text{if } i > 1 \text{ and } j > 1 \\ \infty & \text{otherwise} \end{cases} \tag{12}$$

Where $d(P_i, Q_j)$ is the Euclidean distance between points P_i and Q_j, and $|P|$ and $|Q|$ are the lengths of the trajectories P and Q respectively. A smaller value means a better fit.

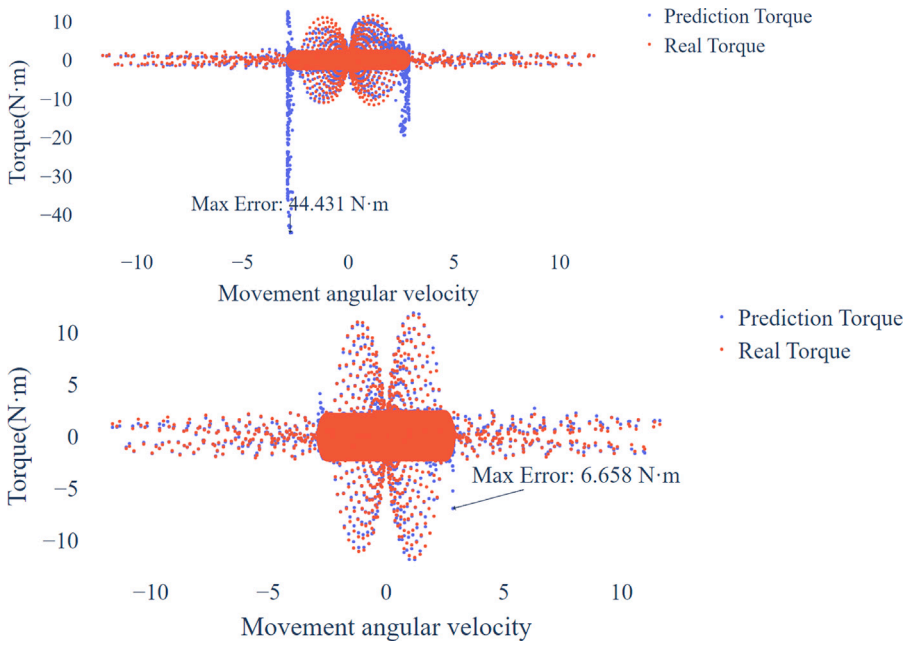


Fig. 11. Comparison of the performance of the DRSN (above) and E2NN (below).

Table 2
Comparison of different methods on various metrics.

Method	Metrics			Response time	Parameters
	MSE	Area Difference	Fréchet Distance		
ANN	0.6	42.9	56.7	3.3×10^{-4}	66 753
E2NN	0.5	8.3	14.3	1.1×10^{-4}	56 223

4.3.3. Time cost

Apart from precision, computational efficiency is also a critical factor that determines the suitability of a dynamics algorithm for real-time applications. Therefore, the evaluation takes into account the expected computation time for each point, and the final assessment will include the average prediction time for a single point.

When assessing the dynamics of robotic arm models, traditional metrics like MSE may not fully capture the nuances of performance. Instead, the Area Difference metric is more informative, offering a holistic view by measuring the cumulative deviation over the entire motion range, which is crucial for systems requiring sustained, smooth operation. Similarly, the Fréchet Distance provides a comprehensive measure of the model's accuracy in following the desired trajectory, by considering the sequence and timing of the arm's motion. These metrics are particularly suited to robotics, where the precision of movement patterns and adherence to the intended path are paramount for practical applications.

4.4. Validation of the E2NN performance

4.4.1. Performance evaluation through simulated data

The model is initially configured for 5000 training iterations. However, implementing an early stop mechanism enabled efficient convergence at 822 iteration epochs, with each step taking between 5–8 ms. This is achieved by training the model in TensorFlow, incorporating early stopping triggered after 60 iteration epochs without improvement, and using a checkpoint system to save the model parameters at the point of minimum training loss. This revised approach ensures a balance between sufficient training and computational efficiency. At training convergence, the torque prediction MAE of the Benchmark DRSN model is 0.10415 and the MAE of E2NN is 0.10716. Using the metrics proposed in Section 4.3, the test results on the 53976 movement recording points are shown in Table 2 and Fig. 11(b). Moreover, the E2NN has a smaller model size and faster response speed than the benchmark DRSN, which makes it a more efficient and practical solution for robot trajectory fitting tasks. The results also show that while the benchmark model and the proposed method have similar error metrics values, the benchmark model's performance is achieved through over-fitting to local points, resulting in a group of large and discrete prediction points around the angular velocity scope in (−5,5). The errors' maximum value is equal to 44.431 N m, which reflects ML's violation of physical facts. In contrast, the proposed E2NN architecture exhibited an overall trend that is closer to the ground truth.

The trajectory fitting results revealed that the E2NN is able to more accurately fit the robot trajectory to the ground truth compared to the benchmark DRSN. The difference in performance between conventional DRSN and E2NN when tested illustrates the existence of over-fitting in DRSN, especially as they perform similarly in training.

The difference between the training and final test results can be attributed to the fact that the training and test data are generated from different simulations. Specifically, the training data are generated from "direct-servo" and "inertia", while the test data are generated from "friction", which covers a wider range of working conditions and has different actual torque functions. However, E2NN is able to compensate for some of the missing information from "simulated training data" by embedding its corresponding equations.

4.4.2. Test E2NN performance on real data

In the current investigation, our objective is to meticulously evaluate and compare a spectrum of ML methodologies in the realm of robot manipulator torque estimation. This evaluation encompasses a strategically selected array of methods, tailored to the context of small sample sizes and complex data structures typical in robotic applications. These include classical machine learning algorithms such as K-Nearest Neighbours (KNN) and Support Vector Machine (SVM), which are renowned

Table 3
Validation on real-world data.

Method	Metric			Maximum error (N m)
	MSE	Area Difference	Fréchet Distance	
Deep MLP	0.00272	3.998	0.249	0.510
SVM	0.575	9.791	1.513	2.263
XGBRegressor	0.00247	3.703	0.223	0.391
KNN	0.00442	4.029	0.209	0.528
DRSN	0.00314	4.835	0.160	0.374
Physics estimation	0.00542	6.519	0.256	0.572
E2NN	**0.00103**	**1.248**	**0.173**	**0.172**

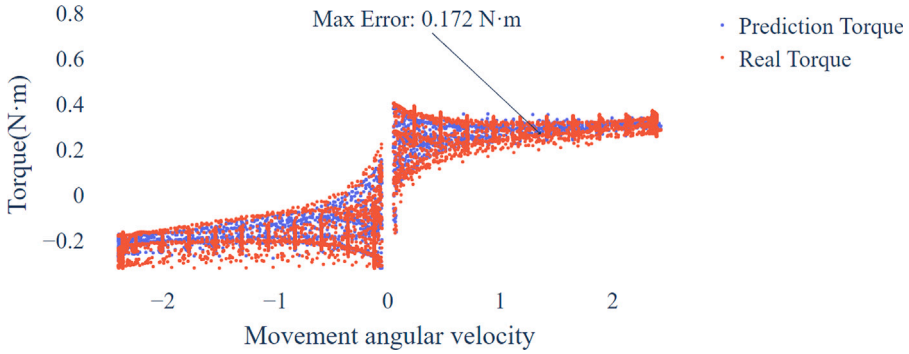


Fig. 12. Prediction results on real robot manipulators.

for their efficacy in smaller datasets. Additionally, we delve into the domain of deep learning, employing Deep Multilayer Perceptron (MLP) and DRSN, both recognized for their profound capabilities in handling high-dimensional data. The XGBRegressor, a gradient-boosted model, is also incorporated due to its proficiency in regression tasks. Furthermore, we explore the application of Nonlinear Regression with Lasso Regularization, where each sub term of the torque equation (referenced as Eq. (8)) functions as an individual operator in physics-based model. Finally, the E2NN-enhanced DRSN, as depicted in Fig. 10, stands as a testament to the integration of PIML into this multifaceted comparative study. Their performance is shown in Table 3 and Figs. 15 to 20.

According to Table 3, it appears that KNN, with an MSE of 0.00442, Area Difference of 4.029, and Fréchet Distance of 0.209, would be a strong contender in this application. However, a deeper analysis reveals that while KNN does outperform SVM (which has a significantly higher MSE of 0.575 and Area Difference of 9.791), it falls short when compared to more advanced methods like Deep MLP and DRSN. The performance gap is particularly noticeable in the context of complex dynamics modelling for robotic arms, where Deep MLPs and DRSNs excel due to their multi-level data representation capabilities. These models, with their advanced feature extraction and noise tolerance abilities, are especially adept at handling the intricate interplay between various input and output variables, such as angular displacement and torque.

Combining Fig. 12 to Fig. 20, which are scatter plots comparing predicted torque to real torque against movement angular velocity, we conclude that E2NN emerges as the most notable model, outperforming all others as it has MSE of 0.00103. It fits the whole trajectory better. Its joint torque predictions have small and few severe deviations from a single large error.

4.5. Discussion of measured data torque prediction results

Building on the discussion of different models' torque prediction results, this paper also investigates the identification of inverse dynamics parameters using the E2NN model, assessing the torque prediction robustness of E2NN in real-world unseen friction scenarios.

4.5.1. Discussion of measured data torque prediction results

In the context of fitting angular displacement, angular velocity, and angular acceleration to robotic arm joint torque, this paper focuses on the performance of different models on such measured small sample data. DRSN stands out among these models due to its balanced approach, offering a low maximum error of 0.374 and the lowest Fréchet Distance of 0.160 among the compared methods. Its architectural advantages, like residual concatenation and contraction operations, contribute significantly to this performance, enhancing feature selection and reducing noise, which is crucial for accurate modelling.

In scenarios with limited data, the XGBRegressor demonstrates its strength. With an MSE of 0.00247 and Area Difference of 3.703, it surpasses the Deep MLP model. This highlights the effectiveness of ensemble methods, particularly in small data contexts. The XGBRegressor, through its boosting process, incrementally corrects previous errors, thereby constructing a robust model that is less prone to overfitting – a common concern in small datasets. Its ability to adapt and improve gradually with each additional model in the ensemble is a key factor in its superior performance.

E2NN outperforms previous models like SVM, KNN, Deep MLP, and DRSN. The combination of the Physics-Informed module, trigonometric functions in the residual blocks, and the equation-based structure of the "Liquid" layer enables the E2NN to effectively mimic the inverse dynamics process of the robotic arm under physical constraints. It leads to notable improvements in predictive accuracy and generalization capabilities over models that do not incorporate such domain knowledge.

Table 4
Results of β estimation for the robotic manipulators torque model.

β_i	1	2	3	4	5	6	7	8	9	10
Real	0	0.600	0	0	0	0.010	0.015	0.200	0.100	0.300
Prosed methods	−0.0132	0.617	0.146	0.023	0.0987	0.0134	0.126	0.303	0.116	0.312

Table 5
Comparison of the robustness of different methods across various metrics when applied to new data.

Method	Metrics			Response time	Parameters
	MSE	Area Difference	Fréchet Distance		
DRSN	1.1	14.4	8.2	1.2×10^{-4}	66 753
E2NN	0.3	1.6	1.6	8.8×10^{-5}	56 223

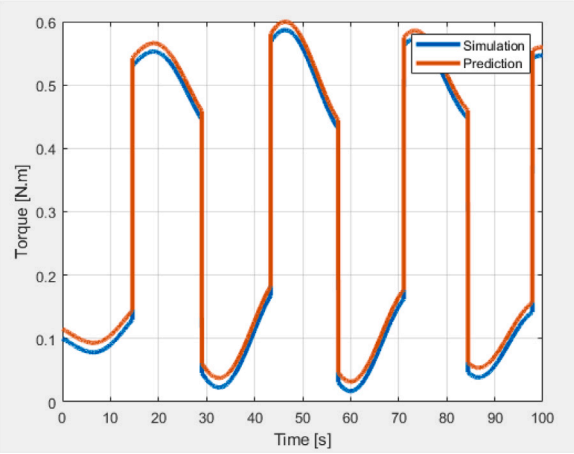


Fig. 13. Reconstructed torque by using Matlab.

One can see that E2NN can accurately fit the trajectory with high precision and computational efficiency under real friction conditions, while also exhibiting a good fit to the robot manipulator's motion. The model's performance is better than the simulation data, which suggests that the impact of friction on the robot manipulators during single-joint motion is less pronounced in the real-world scenario than in the simulation.

4.5.2. Identification of inverse dynamics parameters

In this paper, E2NN outputs joint torque, and its embedded physical operator can also serve to estimate the joint parameter β. To evaluate the E2NN's learning mechanism and its grasp of robot motion dynamics, we extracted the weights corresponding to the embedded operator's NN layer and compared them with the β values from the generated simulation data. This comparison allowed us to assess whether E2NN has effectively learned the robot motion dynamics.

The average weights of the E2NN dense layer are used as β values for the reconstructed robot manipulator's torque trajectory. The fitting performance is evaluated by comparing the reconstructed trajectory with the ground truth trajectory. The obtained results are shown in Table 4 and Fig. 13. The average accuracy of the proposed methods, measured as the MAE, for the parameter estimation in the robotic manipulators torque model is approximately 0.05433. This value represents the average deviation of the estimated values from the actual values across all parameters. The reconstructed trajectories using E2NN's weights have a high torque fitting accuracy reaching 97.1% (as its trajectory is shown in Fig. 14).

These results indicate that the E2NN can approximately simulate the actual behaviour of the robot manipulator. This accuracy is critical in applications where fine motion control and subtle manipulation capabilities are required. The results successfully demonstrate the capability of the E2NN in predicting the joint torques and estimating the joint parameters β of a robotic manipulator. The key innovation of the E2NN lies in its embedded physical operator, which bridges the gap between theoretical modelling of robotics and practical applications.

4.5.3. Investigation on the E2NN's robustness

This paper employs data generated under the "Friction" working condition to evaluate the robustness of the E2NN. This process involves applying the benchmark model and the E2NN, which have been trained on the same dataset, directly to the new test without any additional training. The size of the new test set is 8996 samples. The prediction results of the two models are presented in Table 5. During the steady-state motion of the robot manipulators in the angular velocity range of −2 to 2, it can be observed that the benchmark DRSN model produces large outlier points and extremely unstable predicted curves. The maximum error of DRSN model is 5.3, which is higher than that of the E2NN, and there are significant outliers in the slewing process around −3 and 3. In addition, the benchmark model shows that the prediction results deviate significantly from the observation in the enlarged view of the entire steady-state movement formation. Besides, the E2NN is still able to fit the actual trajectory with promised trend tracking.

5. Conclusion and future works

This study presents the Equation Embedded Neural Network (E2NN), an innovative Physics-informed machine learning approach tailored for addressing robotic inverse dynamics challenges. E2NN showcases notable precision in predicting torque and demonstrates rapid computation capabilities, even with limited joint data. Its effectiveness is corroborated through both simulated and real-world experiments on a 7-degree-of-freedom robotic manipulator. Key contributions of this study include a comprehensive examination of existing hybrid models that merge physical principles with machine learning, the development and assessment of the E2NN framework, the incorporation of a physics regulator to enhance parameter learning, and the creation of a dynamic 'liquid' mechanism for continuous model adaptation.

It is founded and explored that at the heart of E2NN is a unique design that utilizes inverse dynamics equations to form specialized neural network layers. These layers are equipped with tailored activation functions and interconnections that mimic physics operators, thus embedding physical knowledge directly into the network architecture. A standout feature of E2NN is its ability to dynamically adjust the connections between layers through gating units. This allows the network to spontaneously generate various combinations of physical operators in response to the input data. This dynamic, fluid-like structure is key to E2NN's adaptability. It allows the network to seamlessly adjust to varying inputs and equations of motion, thereby maintaining a dynamic representation of physics knowledge. This innovative approach marks a significant advancement in the field of Physics-informed machine learning and opens new avenues for robotic control and simulation.

For the future development of the proposed methodology, E2NN framework has complexity in its model design. It features custom layers for mathematical combinations, a physical information layer, a liquid mechanism for inter-layer connections, and a loss function with neural network structural parameters and polynomial weights. Some more easy establishment methods are needed. Architecturally, E2NN on this condition can be generalized by sharing parameters across multiple sub-networks. It also exhibits imprecision in estimating inverse dynamics parameters, including non-zero estimates for expected zero parameters, suggesting potential over-parameterization. Incorporating

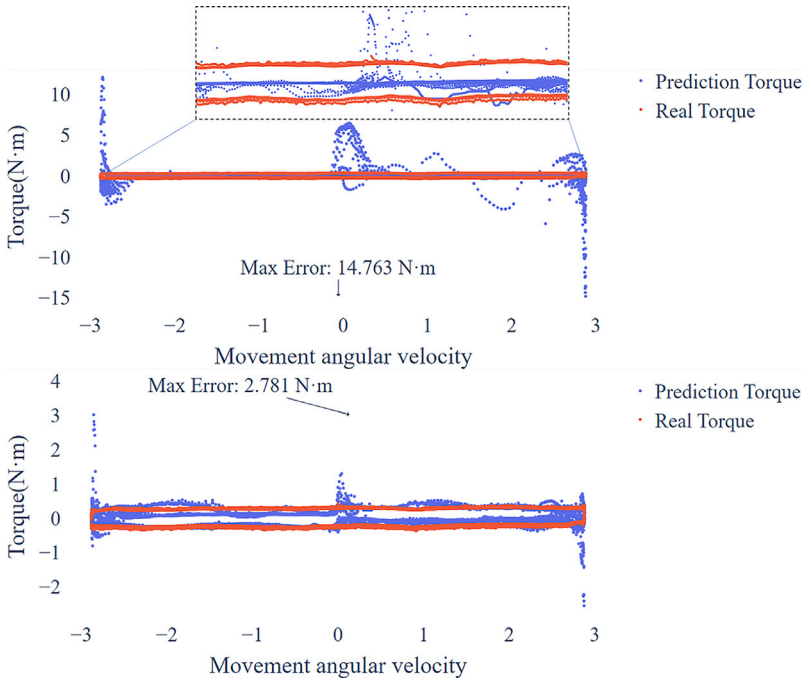


Fig. 14. Comparison between the robustness performance of the E2NN (below) model and the benchmark DRSN (above).

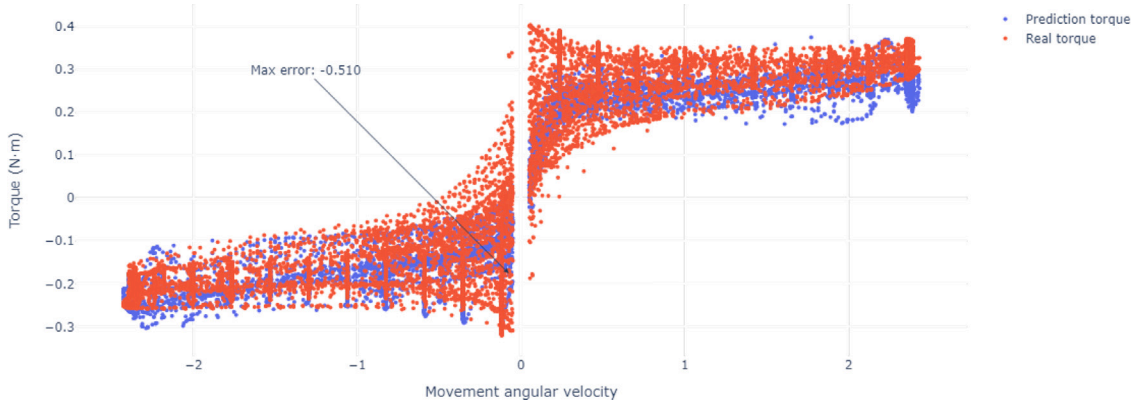


Fig. 15. Torque trajectory predicted by MLP.

prior knowledge of the number and states of pending parameters may be necessary. From the perspective of application, Future work could focus on testing the method on more complex systems, as could be the entire 7-dof manipulator and collaborative applications where humans and external forces are applied on the robot.

CRediT authorship contribution statement

Weikun Deng: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Fabio Ardiani:** Writing – original draft, Validation, Formal analysis, Data curation. **Khanh T.P. Nguyen:** Writing – review & editing, Supervision, Investigation, Funding acquisition. **Mourad Benoussaad:** Writing – review & editing, Supervision, Funding acquisition. **Kamal Medjaher:** Writing – review & editing, Supervision, Project administration, Investigation, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Annex

See Figs. 15–20.

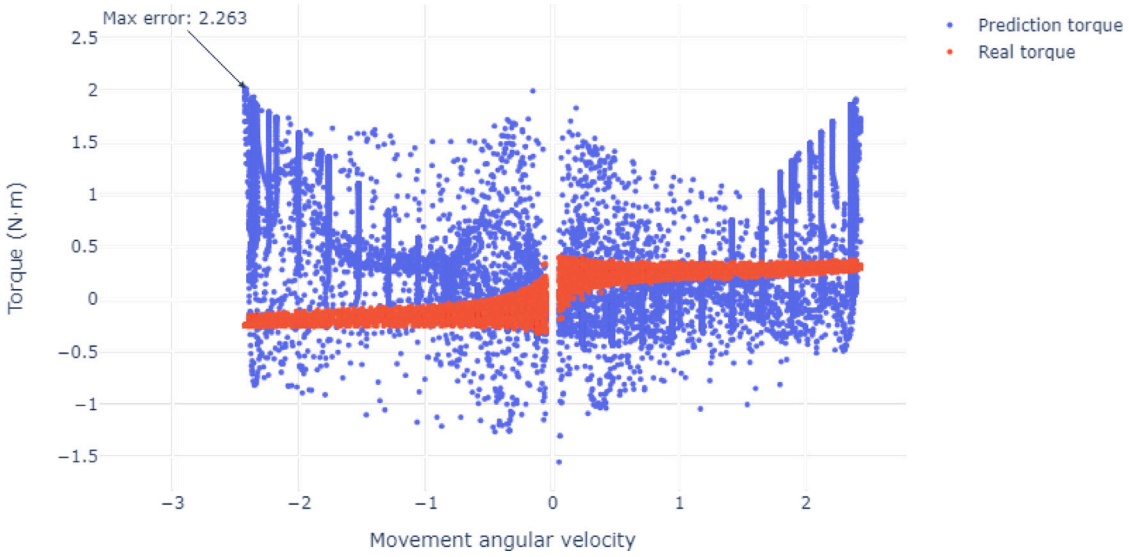


Fig. 16. Torque trajectory predicted by SVM.

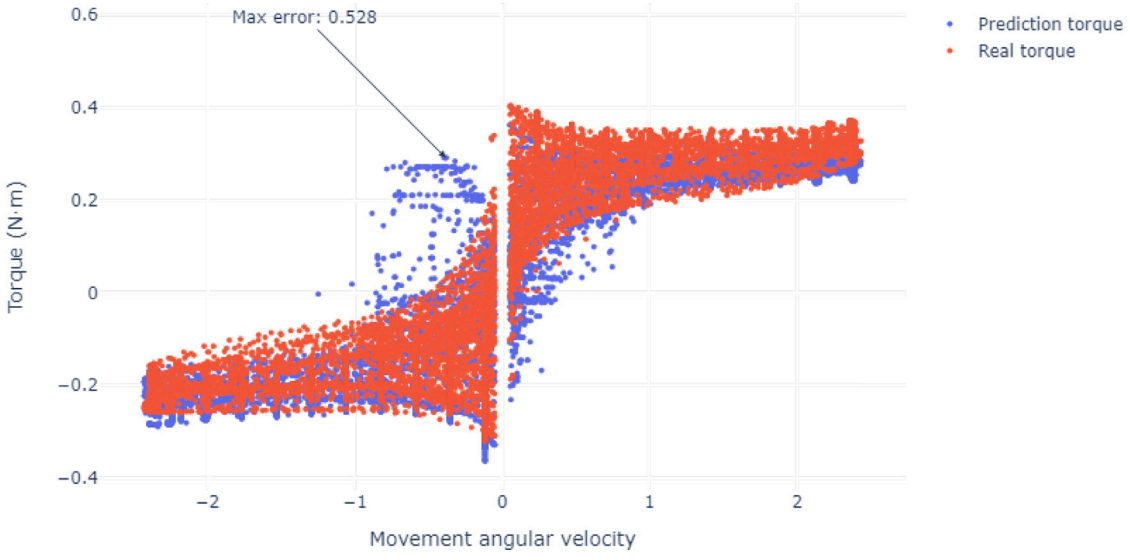


Fig. 17. Torque trajectory predicted by KNN.

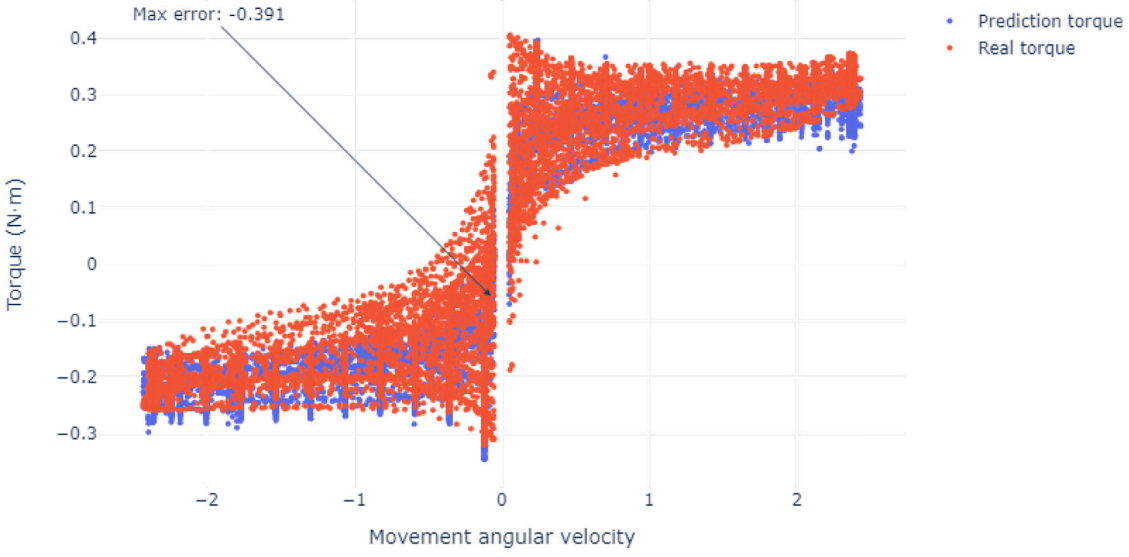


Fig. 18. Torque trajectory predicted by XGBRegressor.

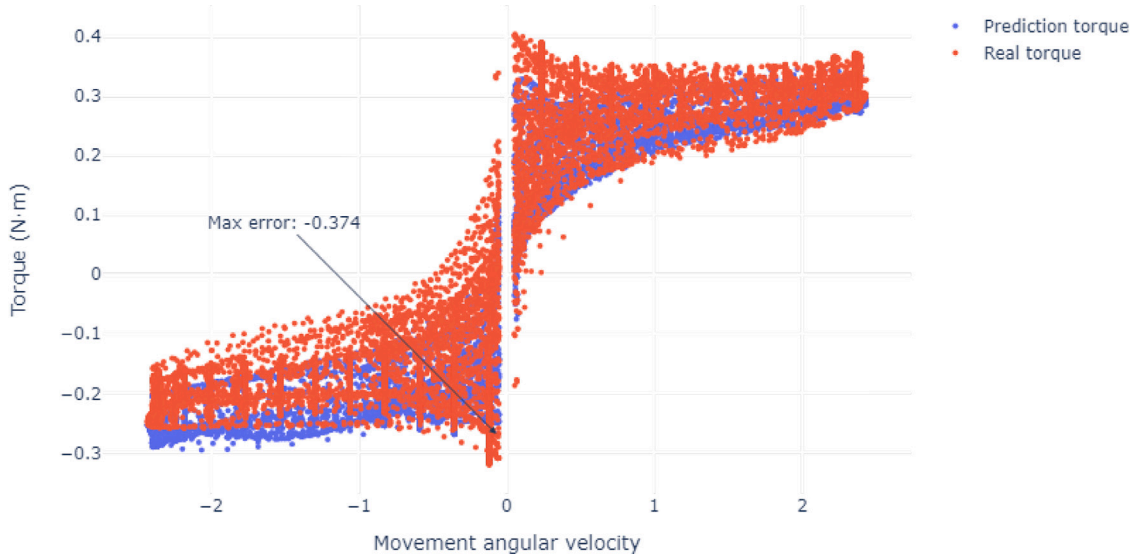


Fig. 19. Torque trajectory predicted by DRSN.

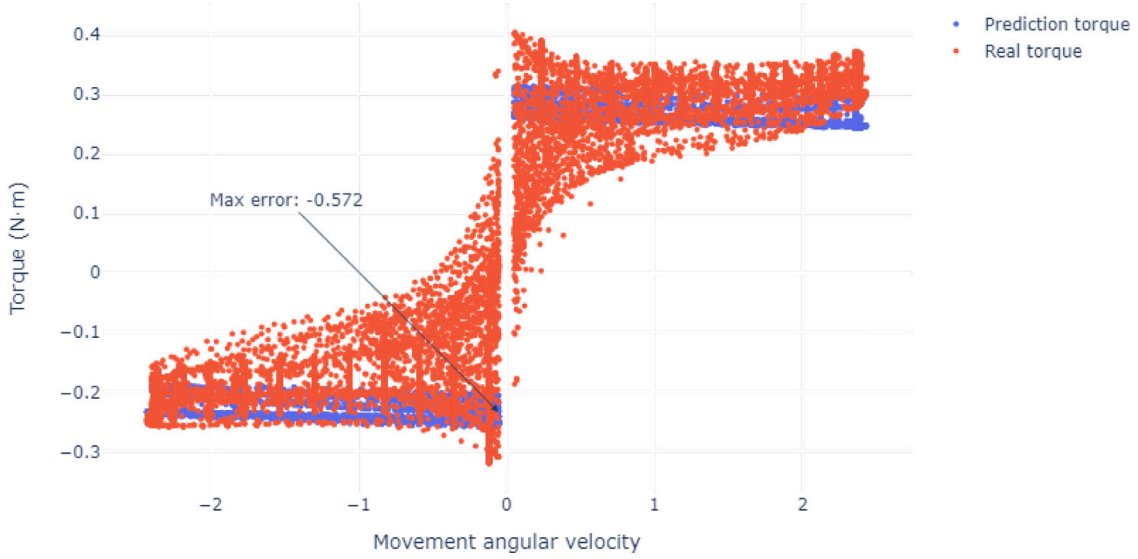


Fig. 20. Torque trajectory predicted by Physics model.

References

[1] S. Moberg, Modeling and Control of Flexible Manipulators (Ph.D. thesis), Linköping University Electronic Press, 2010.

[2] B. Siciliano, L. Sciavicco, L. Villani, G. Oriolo, Robotics: Modelling, Planning and Control, Springer Science & Business Media, 2010.

[3] B. Armstrong, O. Khatib, J. Burdick, The explicit dynamic model and inertial parameters of the PUMA 560 arm, in: IEEE Proceedings. International Conference on Robotics and Automation, Vol. 3, 1986, pp. 510–518.

[4] A.K. Tangirala, Principles of System Identification: Theory and Practice, Crc Press, 2018.

[5] F. Ardiani, Contribution to the Parametric Identification of Dynamic Models: Application to Collaborative Robotics (Ph.D. thesis), Toulouse, ISAE, 2023.

[6] Q. Leboutet, J. Roux, A. Janot, J.R. Guadarrama-Olvera, G. Cheng, Inertial parameter identification in robotics: A survey, Appl. Sci. 11 (9) (2021) 4303.

[7] W. Khalil, E. Dombre, Modeling Identification and Control of Robots, CRC Press, 2002.

[8] S. Briot, M. Gautier, Global identification of joint drive gains and dynamic parameters of parallel robots, Multibody Syst. Dyn. 33 (1) (2015) 3–26.

[9] A. Fabio, B. Mourad, A. Janet, On the dynamic parameter identification of collaborative manipulators: Application to a KUKA iiwa, in: 2022 17th International Conference on Control, Automation, Robotics and Vision, ICARCV, IEEE, 2022, pp. 468–473.

[10] A. Janot, P.-O. Vandanjon, M. Gautier, A generic instrumental variable approach for industrial robot identification, IEEE Trans. Control Syst. Technol. 22 (1) (2013) 132–145.

[11] F. Ardiani, M. Benoussaad, A. Janot, Comparison of least-squares and instrumental variables for parameters estimation on differential drive mobile robots, IFAC-PapersOnLine 54 (7) (2021) 310–315.

[12] M. Gautier, A. Janot, P.-O. Vandanjon, A new closed-loop output error method for parameter identification of robot dynamics, IEEE Trans. Control Syst. Technol. 21 (2) (2012) 428–444.

[13] M. Brunot, A. Janot, F. Carrillo, H. Garnier, Comparison between the IDIM-IV method and the DIDIM method for industrial robots identification, in: IEEE International Conference on Advanced Intelligent Mechatronics, AIM, 2017, pp. 571–576.

[14] M. Brunot, A. Janot, F. Carrillo, J. Cheong, J.-P. Noël, Output error methods for robot identification, J. Dyn. Syst. Meas. Control 142 (3) (2020) 031002.

[15] F. Ardiani, M. Benoussaad, A. Janot, Improving recursive dynamic parameter estimation of manipulators by knowing robot's model integrated in the controller, IFAC-PapersOnLine 55 (20) (2022) 223–228.

[16] A. Bahloul, S. Tliba, Y. Chitour, Dynamic parameters identification of an industrial robot with and without payload, Ifac-Papersonline 51 (15) (2018) 443–448.

[17] A. Fabio, A. Janot, B. Mourad, Industrial robot parameter identification using a constrained instrumental variable method, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2022, pp. 6250–6255.

[18] C.A. Lightcap, S.A. Banks, An extended Kalman filter for real-time estimation and control of a rigid-link flexible-joint manipulator, IEEE Trans. Control Syst. Technol. 18 (1) (2009) 91–103.

[19] M. Ruderman, Modeling of elastic robot joints with nonlinear damping and hysteresis, Robot Syst. Control Prog. (2012) 293–312.

[20] M. Indri, S. Trapani, Framework for static and dynamic friction identification for industrial manipulators, ASME Trans. Mechatron. 25 (3) (2020) 1589–1599.

[21] S. Surati, S. Hedaoo, T. Rotti, V. Ahuja, N. Patel, Pick and place robotic arm: a review paper, Int. Res. J. Eng. Technol. 8 (2) (2021) 2121–2129.

[22] A. Mousaei, M. Gheisarnejad, M.H. Khooban, Robust sliding mode control for two-wheel robot without kinematic equations, 2023.

[23] P. Moradi, M.H. Korayem, N.Y. Lademakhi, Online identification and robust compensation of extended nonlinear time-varying friction model in robotic arms, J. Mech. Sci. Technol. 37 (1) (2023) 367–373.

[24] J.A. Luz Jr., J.M. Balthazar, M.A. Ribeiro, F.C. Janzen, A.M. Tusset, Dynamic model of a robotic manipulator with one degree of freedom with friction component, Int. J. Robotics Control Syst. 3 (2) (2023).

[25] R. Mukhopadhyay, R. Chaki, A. Sutradhar, P. Chattopadhyay, Model learning for robotic manipulators using recurrent neural networks, in: TENCON 2019 - 2019 IEEE Region 10 Conference, TENCON, 2019, pp. 2251–2256, http://dx.doi.org/10.1109/TENCON.2019.8929622.

[26] F. Semeraro, A. Griffiths, A. Cangelosi, Human-robot collaboration and machine learning: A systematic review of recent research, Robot. Comput.-Integr. Manuf. 79 (2022).

[27] L. Jin, S. Li, B. Hu, M. Liu, A survey on projection neural networks and their applications, Appl. Soft Comput. 76 (2019) 533–544.

[28] L. Ljung, Q. Zhang, P. Lindskog, A. Juditski, Estimation of grey box and black box models for non-linear circuit data, IFAC Proc. Vol. 37 (13) (2004) 399–404.

[29] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.

[30] B.S. Pavse, F. Torabi, J. Hanna, G. Warnell, P. Stone, Ridm: Reinforced inverse dynamics modeling for learning from a single observed demonstration, IEEE Robot. Autom. Lett. 5 (4) (2020) 6262–6269.

[31] M. Lutter, J. Peters, Combining physics and deep learning to learn continuous-time dynamics models, Int. J. Robotics Res. 42 (3) (2023) 83–107.

[32] X. Wang, X. Liu, L. Chen, H. Hu, Deep-learning damped least squares method for inverse kinematics of redundant robots, Measurement 171 (2021) 108821.

[33] F. Djeumou, C. Neary, E. Goubault, S. Putot, U. Topcu, Neural networks with physics-informed architectures and constraints for dynamical systems modeling, in: Learning for Dynamics and Control Conference, PMLR, 2022, pp. 263–277.

[34] W. Sun, N. Akashi, Y. Kuniyoshi, K. Nakajima, Physics-informed recurrent neural networks for soft pneumatic actuators, IEEE Robot. Autom. Lett. 7 (3) (2022) 6862–6869.

[35] H. Ren, P. Ben-Tzvi, Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks, Robot. Auton. Syst. 124 (2020) 103386.

[36] N. Yilmaz, J.Y. Wu, P. Kazanzides, U. Tumerdem, Neural network based inverse dynamics identification and external force estimation on the da Vinci research kit, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 1387–1393, http://dx.doi.org/10.1109/ICRA40945.2020.9197445.

[37] M. Lahariya, C. Innes, C. Develder, S. Ramamoorthy, Learning physics-informed simulation models for soft robotic manipulation: A case study with dielectric elastomer actuators, in: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2022, pp. 11031–11038.

[38] C. Rodwell, P. Tallapragada, Physics-informed reinforcement learning for motion control of a fish-like swimming robot, 2022.

[39] K. Morse, N. Das, Y. Lin, A.S. Wang, A. Rai, F. Meier, Learning state-dependent losses for inverse dynamics learning, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2020, pp. 5261–5268.

[40] F. Cursi, D. Chappell, P. Kormushev, Augmenting loss functions of feedforward neural networks with differential relationships for robot kinematic modelling, in: 2021 20th International Conference on Advanced Robotics, ICAR, IEEE, 2021, pp. 201–207.

[41] G. Pizzuto, M. Mistry, Physics-penalised regularisation for learning dynamics models with contact, in: Learning for Dynamics and Control, PMLR, 2021, pp. 611–622.

[42] S. Dereli, R. Köker, A meta-heuristic proposal for inverse kinematics solution of 7-DOF serial robotic manipulator: quantum behaved particle swarm algorithm, Artif. Intell. Rev. 53 (2020) 949–964.

[43] R. Ram, P.M. Pathak, S. Junco, Inverse kinematics of mobile manipulator using bidirectional particle swarm optimization by manipulator decoupling, Mech. Mach. Theory 131 (2019) 385–405.

[44] S. Dereli, R. Köker, Simulation based calculation of the inverse kinematics solution of 7-DOF robot manipulator using artificial bee colony algorithm, SN Appl. Sci. 2 (2020) 1–11.

[45] M. Alebooyeh, R.J. Urbanic, Neural network model for identifying workspace, forward and inverse kinematics of the 7-DOF YuMi 14000 ABB collaborative robot, IFAC-PapersOnLine 52 (10) (2019) 176–181.

[46] J.S. Toquica, P.S. Oliveira, W.S. Souza, J.M.S. Motta, D.L. Borges, An analytical and a deep learning model for solving the inverse kinematic problem of an industrial parallel robot, Comput. Ind. Eng. 151 (2021) 106682.

[47] J. Demby's, Y. Gao, G.N. DeSouza, A study on solving the inverse kinematics of serial robots using artificial neural network and fuzzy neural network, in: 2019 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE, IEEE, 2019, pp. 1–6.

[48] D. Romeres, M. Zorzi, R. Camoriano, S. Traversaro, A. Chiuso, Derivative-free online learning of inverse dynamics models, IEEE Trans. Control Syst. Technol. 28 (3) (2019) 816–830.

[49] P.O. Sturm, A.S. Wexler, Conservation laws in a neural network architecture: enforcing the atom balance of a Julia-based photochemical model (v0. 2.0), Geosci. Model Dev. 15 (8) (2022) 3417–3431.

[50] M. Gautier, W. Khalil, Direct calculation of minimum set of inertial parameters of serial robots, IEEE Trans. Robotics Autom. 6 (3) (1990) 368–373.

[51] W. Khalil, F. Bennis, Comments on" direct calculation of minimum set of inertial parameters of serial robots", IEEE Trans. Robot. Autom. 10 (1) (1994) 78–79.

[52] M. Gautier, Numerical calculation of the base inertial parameters of robots, J. Robotic Syst. 8 (4) (1991) 485–506.

[53] G.H. Golub, C.F. Van Loan, Matrix computations, JHU Press, 2013.

[54] J. Santolaria, M. GinéS, Uncertainty estimation in robot kinematic calibration, Robot. Comput.-Integr. Manuf. 29 (2) (2013) 370–384.

[55] L. Ljung, System Identification: Theory for the User, Pearson Education, 1998.

[56] F. Pukelsheim, Optimal Design of Experiments, SIAM, 2006.

[57] J. Swevers, W. Verdonck, J. De Schutter, Dynamic model identification for industrial robots, IEEE Control Syst. Mag. 27 (5) (2007) 58–71.

[58] J. Jia, M. Zhang, X. Zang, H. Zhang, J. Zhao, Dynamic parameter identification for a manipulator with joint torque sensors based on an improved experimental design, Sensors 19 (10) (2019) 2248.

[59] L. Simoni, M. Beschi, G. Legnani, A. Visioli, Friction modeling with temperature effects for industrial robot manipulators, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, 2015, pp. 3524–3529.

[60] A. Raviola, R. Guida, A. De Martin, S. Pastorelli, S. Mauro, M. Sorli, Effects of temperature and mounting configuration on the dynamic parameters identification of industrial robots, Robotics 10 (3) (2021) 83.

[61] P. Hamon, M. Gautier, P. Garrec, A. Janot, Dynamic identification of robot with a load-dependent joint friction model, in: 2010 IEEE Conference on Robotics, Automation and Mechatronics, IEEE, 2010, pp. 129–135.

[62] Y. Dai, J. Wang, J. Li, Dynamic environment prediction on unmanned mobile manipulator robot via ensemble convolutional randomization networks, Appl. Soft Comput. 125 (2022) 109136.

[63] R. Hasani, M. Lechner, A. Amini, D. Rus, R. Grosu, Liquid time-constant networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, No. 9, 2021, pp. 7657–7666.

[64] G. Schreiber, A. Stemmer, R. Bischoff, The fast research interface for the kuka lightweight robot, in: IEEE Workshop on Innovative Robot Control Architectures for Demanding (Research) Applications how To Modify and Enhance Commercial Controllers, ICRA, 2010, pp. 15–21.

[65] L. Tonin, F.C. Bauer, J.d.R. Millán, The role of the control framework for continuous teleoperation of a brain–machine interface-driven mobile robot, IEEE Trans. Robot. 36 (1) (2019) 78–91.

[66] H. Shen, L. Pan, J. Qian, Research on large-scale additive manufacturing based on multi-robot collaboration technology, Addit. Manuf. 30 (2019) 100906.