Physics-Informed Neural Network for Model Prediction and Dynamics Parameter Identification of Collaborative Robot Joints

Xingyu Yang, *Graduate Student Member, IEEE*, Yixiong Du, Leihui Li, Zhengxue Zhou, and Xuping Zhang, *Member, IEEE*

***Abstract*—Collaborative robots have promising potential for widespread use in small-and-medium-sized enterprise (SME) manufacturing and production due to the development of increasingly sophisticated Human-Robot Collaboration technologies. However, predicting and identifying the behavior of collaborative robots remains a challenging problem due to the significant non-linear properties of their unique gearbox, the harmonic drive. To tackle the engineering problem, this work proposes a physics-informed neural network (PINN) to predict and identify collaborative robot joint dynamics. The procedure involves deriving the state-space dynamic model, embedding the system's dynamics into a recurrent neural network (RNN) with customized Runge-Kutta cells, obtaining labeled training data, predicting system responses, and estimating dynamic parameters. The proposed method is applied to predict and identify collaborative robot joint dynamics, and the results are verified and validated through numerical simulations and experimental testing, respectively. The obtained results demonstrate a high level of agreement with the ground truth and exhibit superior performance compared to the conventional PINN and the non-linear grey-box state-space estimation algorithm when confronted with non-linearity and dynamic coupling. Moreover, the PINN exhibits the potential for extension to various dynamic systems.**

***Index Terms*—Calibration and Identification, Deep Learning Methods, Dynamics, Actuation and Joint Mechanisms, Human-Robot Collaboration.**

I. Introduction

COLLABORATIVE robots are a key feature of future small-and-medium-sized enterprise (SME) manufacturing and production. Accurately controlling a collaborative robot is always a top priority [1], as it requires precise dynamic modeling of the robot, interaction with humans or environments [2], [3], intelligent control strategies to adapt to various tasks [4], hardware innovation to increase the robot's perception of uncertainties in the environment, and active learning algorithms to update its production skills [5]. Of these requirements, the precise dynamic modeling of collaborative robots is still a hot topic after decades of research [6], [7], [8], [9]. This is due to the unique and highly integrated joint design of collaborative robots, which introduces challenges to the dynamic model structure and parameter calibration because of their non-negligible joint flexibility. One of the main components inside a collaborative robot is the gear reducer or harmonic drive, which is different from conventional gearboxes in both stiffness and transmission principles. As a result, an apparent non-linearity can be observed in collaborative robots [10], [11]. Moreover, the primary control of the robot is still performed in joint space by transforming from task space to joint space with Jacobian Matrix. Therefore, to better control the robot's performance, it is essential to develop a dynamic model of the robot joint and to predict and identify the system with these nonlinear behaviors [7], [8].

Model prediction and parameter identification are commonly referred to as forward and inverse problems, respectively. The forward problem involves mapping the system input and output using available data, while the inverse problem focuses on discovering the system's underlying structure and parameters based on the collected data [12]. The inverse problem, also called system identification, is a traditional topic in control that has been explored for over half a century. The final goal of system identification is to infer the model parameters based on the evaluation of the collected experimental data [13]. By choosing or developing the appropriate model structure to construct the regression model, the regression algorithms are then applied to minimize the objective function and obtain the estimated values of the parameters, empowering the model to map the inputs to outputs as the actual system. While conventional system identification algorithms have demonstrated high efficiency in linear systems, their application in nonlinear systems remains an active area of research [14], [15]. In recent years, machine learning techniques have gained significant attention in the field of modeling and estimation [16], [17]. A conclusion that can be drawn from existing research is that black box machine learning models are efficient tools to address the forward problem and replace the traditional physical law-based models if only the relationship between system inputs and outputs is desired. This could have

Manuscript received 5 July 2023; accepted 25 October 2023. Date of publication 2 November 2023; date of current version 13 November 2023. This letter was recommended for publication by Associate Editor U-X. Tan and Editor L. Pallottino upon evaluation of the reviewers' comments. The work of Xingyu Yang was supported in part by China Scholarship Council. This work was supported in part by the Danish Ministry of Higher Education and Science, in part by Bilateral Research Networking in Customizable Collaborative Industrial Robots for SME Manufacturing, in part by the European Union Regional Fund, and in part by Integrating Human-Robot Collaboration into Danish SME Manufacturing and Production based on the collaboration among Jydsk Emblem Fabrik A/S, NIZE Equipment A/S, and Aarhus University. *(Corresponding author: Xuping Zhang.)*

Xingyu Yang, Yixiong Du, Leihui Li, and Xuping Zhang are with the Department of Mechanical and Production Engineering, Aarhus University, 8200 Aarhus N, Denmark (e-mail: xingyu_yang@mpe.au.dk; dyx2021@outlook.com; leihui@mpe.au.dk; xuzh@mpe.au.dk).

Zhengxue Zhou is with the Leverhulme Research Centre for Functional Materials Design, University of Liverpool, L73NY Liverpool, U.K. (e-mail: z.z.zhou@liverpool.ac.uk).

Digital Object Identifier 10.1109/LRA.2023.3329620

many benefits for model prediction and system control [18], [19], [20]. However, conventional neural networks are often treated as black boxes, lacking explicit physical interpretations for their neurons. Moreover, the complex mathematical mapping makes it challenging to represent this conventional machine learning model more intuitively as physical laws, making it difficult to infer the physical system parameters and provide guidance on designing the system. In addition, conventional supervised learning algorithms have a strong dependence on the quality and amount of data, which further increases the difficulties of training the model to represent model dynamics [12].

The traditional physical law-based modeling approach still offers irreplaceable advantages in terms of simplifying the model while maintaining high precision [21], [22]. Furthermore, it is less dependent on data. Therefore, the concept of Physics-Informed Neural Networks (PINNs) [12], [23] has been proposed and developed by combining physical knowledge with advanced machine learning models, where the physical laws are colocated to the neural networks to serve as constraints. As a result, it has the benefits of both physical and machine learning models and overcomes their respective shortcomings. For instance, the machine learning models excel in regression tasks and can effectively handle uncertainties, while the incorporation of physical laws reduces the reliance on extensive data and provides valuable guidance for optimizing the loss function. The PINNs were first proposed to solve partial differential equations and identify system coefficients in both continuous and discrete-time systems, such as the Schroedinger equation and Allen-Cahn equation [12]. Subsequently, the PINN has gained much attention from the research community and has been widely applied to fluid dynamics [24], chemical kinetics [25], and mechanics [26], among others. PINNs perform as well as other machine learning algorithms for model prediction [27] and and have excellent performance in identifying model parameters due to their transparency [28], [29], [30]. Moreover, some Python libraries, such as DeepXDE [31], which are based on the original concept of PINNs, have been released and have facilitated the application of PINNs across various domains.

In a nutshell, the conventional PINN [32] involves a two-stage process without modifying the neural network structure. Instead, additional trainable hyperparameters are only assigned to represent the physical system parameters in the loss function computation. The neural network acts as a general function approximator and is typically a conventional neural network. However, this approach may face challenges when dealing with complex systems that have a large number of parameters, particularly systems governed by non-linear and dynamically coupled ordinary differential equations (ODEs), due to the simplicity of the network structure. While PINNs have been extensively explored in fluid dynamics applications [33], their application to mechanical systems has mainly been limited to linear systems [29], [30]. Therefore, there is a pressing demand to extend the concept of PINNs to complex ODEs systems with non-linear characteristics, especially in the context of joint dynamics, where precise model prediction and parameter identification pose significant research challenges. Therefore, the main objective of this work is to address the challenges by revising the structure of conventional PINNs from a two-stage approach to a hybrid form (H-PINN), where the physical laws are embedded into the neural network, to enhance the capability of PINNs in handling non-linear ODE systems.

The remainder of this paper is organized as follows: Section II outlines the pipeline of using H-PINN for model prediction and parameter identification, Section III describes the principles and construction of the H-PINN in detail, Section IV explains the parametric modeling process for collaborative robot joint dynamics, Section V presents the acquisition of labeled data for training, Section VI demonstrates the prediction and identification results obtained using H-PINN, and Section VII concludes the paper and discusses future work.

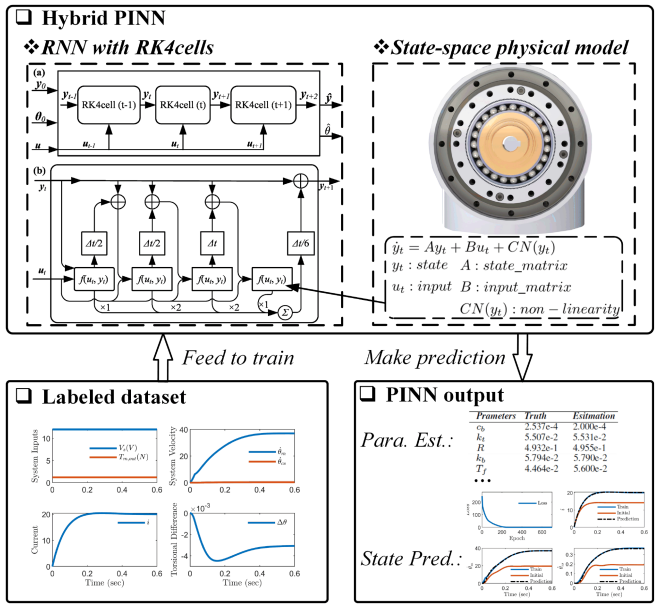

Fig. 1. Framework of using the H-PINN for model prediction and parameter identification following the pipeline of a typical supervised learning.

II. Pipeline of Prediction and Identification With H-PINN

In this section, a framework is presented that follows the pipeline of typical supervised learning to address the problem of model prediction and parameter identification by utilizing the physics-informed neural network, as shown in Fig. 1. The framework comprises three main parts: 1) A labeled dataset for training the network; 2) A hybrid PINN implemented on the RNN with customized 4th order Runge-Kutta (RK4) cells, which will be detailed in the next section; and 3) The outcomes of the H-PINN, which are the estimation of system parameters and prediction of model behaviors.

The labeled dataset can be acquired from physical experiments or numerical simulations, with experimental data being preferred as it can best resemble the real system behaviors. However, the simulated data can also be used to verify whether the machine learning model can capture the ground truths of the system, while the ground truth parameter value of a physical system is not available most of the time. Regarding the H-PINN, the RNN is chosen to serve as the backbone by replacing the cells with customized RK4 cells. The physical law, which represents the state space system dynamic model, is embedded into the cells to reveal the relationship among the system inputs, current states, and previous states. The non-linearity can be modeled and coupled to each state with trainable weights, enabling the H-PINN to be extended to both linear and nonlinear systems. After adequate training, the system dynamics can be estimated parametrically and corresponding model predictions can be made using the H-PINN. The H-PINN offers several advantages over

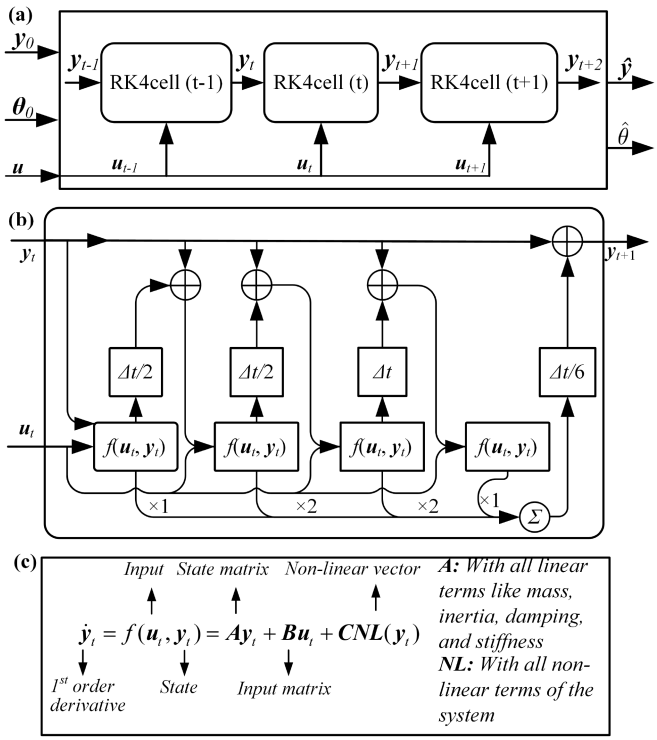

Fig. 2. (a) The proposed H-PINN implemented on RNN with RK4 cells, (b) the detail structure of the RK4 cell, and (c) the interpretation of physics of the RK4 cell.

the conventional system identification approach [34], including its ability to estimate unknown states as if the required ground truths are appropriately labeled. Additionally, after obtaining the dynamic model, the construction of the parameter vector and regressor matrix is not required by the H-PINN.

III. Construction of H-PINN

The RNN offers advantages in capturing sequential features in input data, where previous states and system inputs will impact subsequent states. This aligns with the state space system dynamics, where the solution is a time series. The RNN also allows for parameter sharing across time steps, reducing the number of model hyper-parameters and enabling assigning physical meaning to those parameters [35]. As such, the proposed H-PINN utilizes the RNN as its backbone. The Runge-Kutta method is widely used to obtain numerical solutions of differential equations with simple equations but high precision, both explicitly and implicitly. Thus, the proposed H-PINN embeds the physical law of the system into RNN using RK4. The implementation of the proposed H-PINN is shown in Fig. 2(a). The neural network takes the system input $\boldsymbol{u}$, initial condition of the system states $\boldsymbol{y}_0$, and the initial guess of the system parameters $\boldsymbol{\theta}_0$ as inputs, while the outputs of the network are the estimation of the system states $\hat{\boldsymbol{y}}$ and system parameters $\hat{\boldsymbol{\theta}}$. In each epoch of the training process, the H-PINN predicts the system states and compares them with the ground truth data to compute the loss function. Subsequently, the loss is utilized to optimize the estimation of the system parameters for the next epoch of training.

The detailed structure of the RK4 cell is depicted in Fig. 2(b) and its mathematical representation is shown in (1).

$$\boldsymbol{y}_{t+1} = \phi(\boldsymbol{u}_t, \boldsymbol{y}_t, \Delta t) \tag{1}$$

where the next state, $\boldsymbol{y}_{t+1}$, is determined by the system input, $\boldsymbol{u}_t$, the current state, $\boldsymbol{y}_t$, and the time increment, Δt. The function operator ϕ, which governs the relationship between the current and next states, is also detailed in Fig. 2(c). The function $f(\boldsymbol{u}_t, \boldsymbol{y}_t)$ corresponds to the embedded physics of the dynamic system. Here, the state-space dynamic model is chosen, but it can theoretically be any form of ODEs. The linear terms of the system, such as time-invariant mass/inertia, viscous friction, and linear stiffness, lie in the state matrix $\boldsymbol{A}$. The input matrix $\boldsymbol{B}$ maps the exogenous inputs of the system to their corresponding states. The non-linearity of the system, for instance, the non-linear stiffness and the Coulomb friction, can be formulated as a non-linear vector $\mathbf{NL}(\boldsymbol{y}_t)$, and $\boldsymbol{C}$ represents the selecting matrix of the non-linearities.

The RK4 cell recursively calls the state equation with the RK4 method to perform the numerical integration for predicting the next state $\boldsymbol{y}_{t+1}$. While the state matrix $\boldsymbol{A}$, input matrix $\boldsymbol{B}$, and non-linear vector $\mathbf{NL}(\boldsymbol{y}_t)$ are constructed with trainable variables, which are assigned with physical meaning. During the training process, the value of those trainable variables will approach the ground truth according to the error between the labeled state data and network prediction (the forward problem), and provide the answer to system discovery (the inverse problem). In terms of the implementation of the H-PINN, the TensorFlow platform (version 2.9.1) and Keras API are used, where the native RNN in Keras is modified by replacing the cells with the RK4 cells shown in Fig. 2(b). During the training process, the values of trainable variables are optimized with the *RMSProp* optimizer through loss back-propagation to approximate the ground truth. Particularly, constraints will be applied to ensure that the mass/inertia terms are positive if they are set as trainable.

In the H-PINN construction process, another crucial aspect to consider is the selection of an appropriate loss function. In the context of parameter identification, the problem is typically treated as a regression problem. The mean square error (MSE) and mean absolute error (MAE) are the two most commonly used loss functions, which have shown great success in various applications. However, for dynamic systems with multiple states, the magnitude of each state's response may differ significantly. If only the MSE loss is employed, the contribution of states with smaller responses may be overlooked since their loss values would be relatively insignificant compared to those of the larger response states. To address this issue, the weight of each state in the loss function is computed as shown in (2) to normalize the contributions from different states.

$$w_i = |y_{\max}/y_{i,\max}| \tag{2}$$

where w_i denotes the weight of each state, y_{max} represents the maximum steady-state value of all states obtained from the labeled data, and $y_{i,max}$ represents the max value of each state. By doing so, smaller response states will not be neglected during the training process. Therefore, the loss function utilized in the proposed H-PINN is the weighted mean square error (WMSE) loss, as shown in (3).

$$wmse = \sum_{i=1}^{n} [w_i(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)]^2/n \tag{3}$$

where n is the number of states. $\hat{\boldsymbol{y}}_i$ and $\boldsymbol{y}_i$ are the prediction of H-PINN and labeled training data for each state, respectively. In addtion, to achieve a faster training process, a learning rate schedule with a piece-wise learning rate decay is employed.

In terms of the network structure, while the cells of the RNN have been replaced with RK4 cells, the overall structure of the network still consists of interconnected processing nodes that are processing and analyzing data. Therefore, although the specific type of processing units has changed, the overall structure and function of the network still meet the definition of a neural network [36].

IV. Dynamic Model of Robot Joint

A typical collaborative robot joint can be divided into three major parts: the control circuits, the brushless DC (BLDC) motor, and the gearbox [6]. This section presents dynamic models of the motor and gearbox of a collaborative robot joint.

A. Dynamic Model of the Motor

The dynamic equations of the BLDC motor can be expressed as follows, based on its working principle:

$$Ri + L\dot{i} + k_b\dot{\theta}_m = V_s \tag{4}$$

$$J_r\ddot{\theta}_m + c_b\dot{\theta}_m - k_t i + \tau_{f,c}sign(\dot{\theta}_m) = -\tau_m \tag{5}$$

where J_r, R, L, and i represent the rotor inertia, resistance, inductance, and armature current, respectively. k_t, k_b, and c_b are the torque constant, back electromotive force constant, and viscous friction coefficient from the bearing, respectively. $\dot{\theta}_m$ represents the velocity of the rotor, $\tau_{f,c}$ stands for Coulomb friction, V_s is the supply voltage on the terminals of the armature, and τ_m represents the output torque of the motor. By choosing the system's state vector as $\boldsymbol{q}_1 = [i, \dot{\theta}_m]^T$, the dynamic equation in (4)-(5) can be expressed in the following state space form, which is referred to as *Model 1*.

$$\dot{\boldsymbol{q}}_1 = \begin{bmatrix} -R/L & -k_b/L \\ k_t/J_r & -c_b/J_r \end{bmatrix} \begin{bmatrix} i \\ \dot{\theta}_m \end{bmatrix} + \begin{bmatrix} 1/L & 0 \\ 0 & 1/J_r \end{bmatrix} \begin{bmatrix} V_s \\ -\tau_m \end{bmatrix} - \tau_{f,c} \begin{bmatrix} 0 \\ sign(\dot{\theta}_m)/J_r \end{bmatrix} \tag{6}$$

B. Dynamic Model of the Gearbox

In addition to the dynamic model of the BLDC motor, the gearbox, or harmonic drive, is another major component that significantly affects the dynamic behavior of robot joints. Due to its unique transmission principle, it introduces nonlinear friction, flexibility, and kinematic error to the joint. One common method to model the dynamics of the harmonic drive is to simplify it as a nonlinear torsional spring and add additional terms to approximate the actual behavior [10], [11], [21]. Therefore, Fig. 3 and (7)–(10) represent the schematic diagram and equations of the dynamic model of the gearbox.

$$\Delta\theta = \theta_m/N - q \tag{7}$$

$$i_{HD} = 1/N + \partial(\theta_{ke})/\partial(\theta_m) \tag{8}$$

$$J_m\ddot{\theta}_m + c_m\dot{\theta}_m + c_1 i_{HD}\Delta\dot{\theta} + k_1 i_{HD}\Delta\theta + k_2 i_{HD}\Delta\theta^3 + \tau_{f,m}sign(\dot{\theta}_m) = \tau_m \tag{9}$$

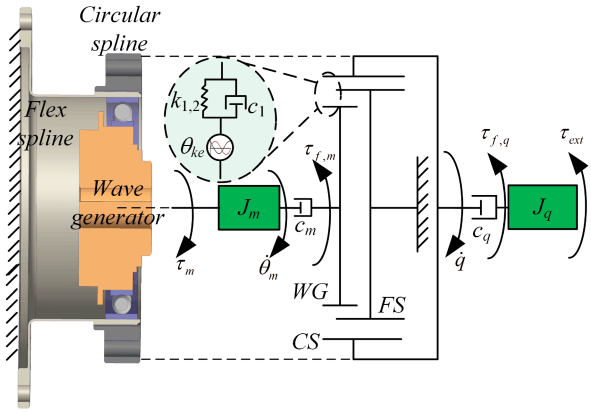

Fig. 3. Simplified dynamic model of harmonic drive in the robot joint.

$$J_q\ddot{q} + c_q\dot{q} - c_1\Delta\dot{\theta} - k_1\Delta\theta - k_2\Delta\theta^3 + \tau_{f,q}sign(\dot{q}) = -\tau_{ext} \tag{10}$$

where τ_m represents the motor torque, J_m is for the concentrated inertia of the motor rotor and wave generator, J_q is the concentrated inertia of the circular spline and load, k_1 and c_1 are the linear torsional spring and damper, while k_2 stands for the nonlinear torsional spring, and θ_{ke} is the kinematic error of the harmonic drive. τ_{ext} and c_q are the external torque and viscous friction coefficient from the load side, respectively. $\Delta\theta$ represents the torsional difference between the input and output shafts, N stands for the reduction ratio, and $\tau_{f,m}$ and $\tau_{f,q}$ are the coulomb friction on the motor and load. To make the dynamic model of the gearbox fit the H-PINN, the state vector $\boldsymbol{q}_2 = [\theta_m, q, \dot{\theta}_m, \dot{q}]^T$ can be chosen to convert it into state space. Then the dynamic model of harmonic drive can be expressed as in (11), which is marked as *Model 2*.

$$\dot{\boldsymbol{q}}_2 = \boldsymbol{A}_2\boldsymbol{q}_2 + \boldsymbol{B}_2\boldsymbol{u}_2 + \boldsymbol{C}_2\boldsymbol{N}(\Delta\dot{\theta}, \Delta\theta, i_{HD}) \tag{11}$$

where $\boldsymbol{A}_2$ is the $4X4$ linear system state matrix, $\boldsymbol{B}_2$ is the $4X2$ input matrix, and $\boldsymbol{N}(\Delta\dot{\theta}, \Delta\theta, i_{HD})$ accounts for friction, hysteresis, and kinematic error. Furthermore, by combining *Model 1* and *Model 2*, the electromechanical model of the collaborative robot joint, marked as *Model 3*, can be expressed in (12). The state vector for this model is chosen as $\boldsymbol{q}_3 = [i, \theta_m, q, \dot{\theta}_m, \dot{q}]^T$.

$$\dot{\boldsymbol{q}}_3 = \boldsymbol{A}_3\boldsymbol{q}_3 + \boldsymbol{B}_3\boldsymbol{u}_3 + \boldsymbol{C}_3\boldsymbol{N}(\Delta\dot{\theta}, \Delta\theta, i_{HD}) \tag{12}$$

V. Data Acquisition

In this section, the acquisition of labeled data for training the H-PINN is discussed, both from simulations of the dynamic model and experiments on the real robot.

A. Forward Dynamic Simulation

To test the performance of the identification algorithm, using numerical simulation data with given model parameters is a common approach. This is because in real experiments, there may be some exceptional situations, such as measurement noise and limitations of the sampling frequency. If the system's nonlinearity is not significant, the noise may cover valuable information and impact the parameter estimation. Also, if the sampling frequency is not high enough, the experiment data

TABLE I
TRAINABLE PARAMETERS OF MODEL 1 AND THEIR ESTIMATIONS WITH THE H-PINN, NLGR AND DEEPXDE (IN SI UNITS)

Θ (Truth)	H-PINN ($e\%$)	NLGR ($e\%$)	DeepXDE ($e\%$)
R (4.932e-1)	**4.937e-1 (0.10)**	**4.937e-1 (0.10)**	4.992e-1 (1.11)
c_b (2.537e-4)	2.382e-4 (6.11)	**2.559e-4 (0.87)**	3.140e-4 (23.7)
k_b (5.794e-2)	**5.795e-2 (0.02)**	**5.793e-2 (0.02)**	5.901e-2 (1.85)
k_t (5.507e-2)	5.481e-2 (0.47)	**5.521e-2 (0.25)**	5.325e-2 (3.30)
τ_{fc} (4.464e-2)	4.711e-2 (5.53)	**4.458e-2 (0.13)**	5.811e-3 (86.9)

TABLE II
PERFORMANCE OF THREE APPROACHES FOR MODEL 1 UNDER DIFFERENT TRAINING CONDITIONS

Method	Epoch	Labels	Samples	Time (s)	MSE
H-PINN	350	2	700	**61.17**	0.0355
	350	2	500	**43.40**	0.0784
	350	1	500	**44.03**	0.1212
DeepXDE	30000	2	700	330.32	0.6759
	30000	2	500	321.24	0.7405
	30000	1	500	256.16	2.6606
NLGR	–	2	700	286.97/12.26	**2.9694e-5**
	–	2	500	284.67/12.37	**2.9694e-5**
	–	1	500	251.34/13.07	**0.0894**

TABLE III
TRAINABLE SYSTEM PARAMETERS OF MODEL 3 AND THEIR ESTIMATIONS WITH THE H-PINN AND NLGR (IN SI UNITS)

Para. Θ	Truth	H-PINN Est. ($e\%$)	NLGR Est. ($e\%$)
R	4.932e-1	**4.968e-1 (0.73)**	5.093e-1 (3.26)
c_b	1.459e-4	**1.561e-4 (6.99)**	4.886e-04 (234.89)
k_b	5.794e-2	**5.451e-2 (5.43)**	4.690e-2 (18.63)
k_t	5.507e-2	**5.631e-2 (2.25)**	5.970e-2 (8.41)
c_m	3.04e-2	**3.144e-2 (3.42)**	3.407e-2 (12.07)
c_1	7.100	**7.621 (7.34)**	2.142e1 (201.69)
c_q	4.031e1	**4.197e1 (4.12)**	2.662e1 (33.96)
k_1	8.000e3	**8.265e3 (3.31)**	7.035e3 (12.06)
k_2	4.000e5	**4.169e5 (4.23)**	6.023e5 (50.58)
$\tau_{f,m}$	4.464e-2	**4.000e-2 (10.03)**	**4.000e-2 (10.03)**

will fail to capture high-frequency features. However, numerical simulations can be considered as measurements with infinite sampling frequency and zero noise. Moreover, one can quickly obtain multiple data sets and test the algorithm's sensitivity in different parameter value ranges. Therefore, the *Model 1* and *Model 3* with pre-defined trainable parameter values listed in Tables I and III are used to obtain their simulated step responses, and the numerical solver *ODE45* is utilized in *MATLAB* to get the simulation data, where Fig. 4 showcase the simulated result of *Model 3*.

B. Experimental Data

Besides the simulated data, experimental data is acquired to validate the effectiveness of the proposed method on system identification. The dynamic response of the collaborative robot joint is tested on a UR3e robotic arm. The setup used for the experiments is shown in Fig. 5. During testing, only the base joint is driven, and motion data from the link side and motor side are recorded by two built-in encoders inside the base joint. The motor torque is estimated using the onboard current sensor, and no external force is applied to the robot during testing. The general masses of the rest five joints collapse into the link inertia of the joint dynamics. To obtain the accelerations of the

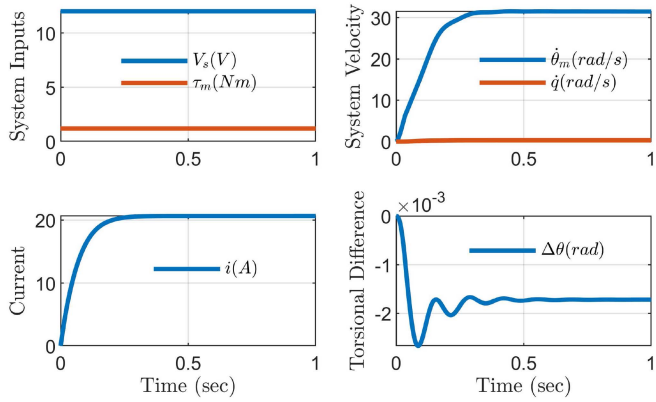

Fig. 4. Numerically simulated step response of *Model 3* with parameters defined in Table III.

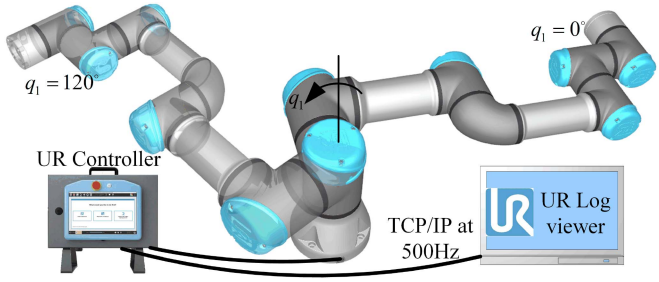

Fig. 5. Experimental testing of dynamic behavior on the UR3e base joint.

motor and link, the velocity is differentiated, and a 4th order Butterworth filter is applied with a cutting frequency of 50 Hz to remove uncertain dynamics and additional noise introduced by the differentiation process [37].

VI. RESULTS AND DISCUSSION

This section presents the performance of the proposed H-PINN, the non-linear grey-box state-space identification method (NLGR) [22], and the conventional two-stage PINN (DeepXDE) [31] in parameter identification and model state prediction.

A. The Perfromance of H-PINN on Numerically Simulated Data

To validate the identification and prediction outcomes of H-PINN, the non-learning-based approach NLGR is implemented using the same state space models in MATLAB to generate comparison results. Additionally, the conventional two-stage PINN is implemented with the DeepXDE library to provide a more comprehensive evaluation of their performance. All approaches are trained using the same data and initial conditions. The numerically simulated response of *Model 1* with a step input of [12,0] is initially used to train H-PINN alongside the corresponding state space model. The prediction results of H-PINN, NLGR, and DeepXDE for *Model 1* are illustrated in Fig. 6. Their parameter identification results are listed in Table I, where the estimation error is calculated as $e = |Est. - Truth|/Truth \times 100\%$.

The left subplot of Fig. 6 shows the training loss of H-PINN for DC motor identification. Since *Model 1* is primarily a linear model with only a discontinuity caused by Coulomb friction, the

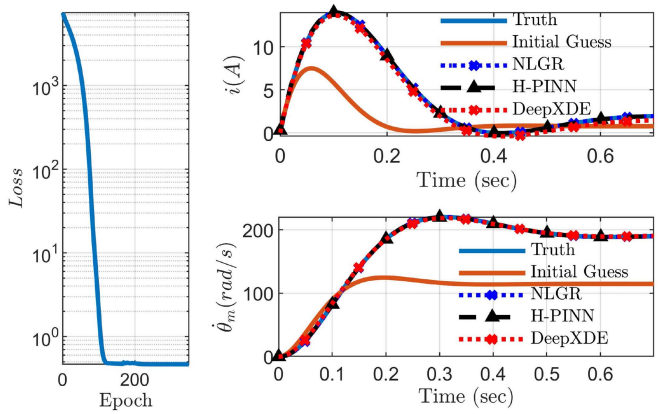

Fig. 6. State predictions of H-PINN, NLGR and DeepXDE for *Model 1*.

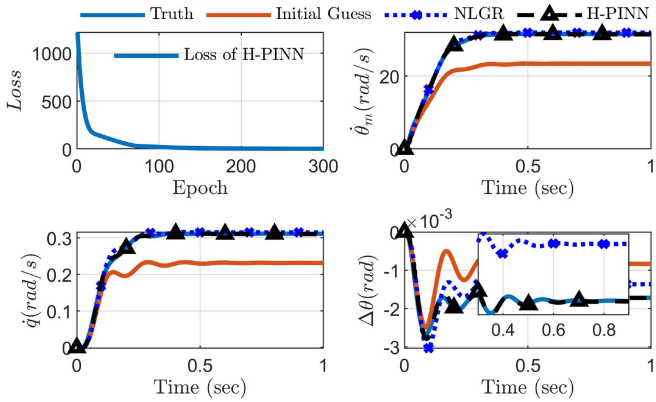

Fig. 7. Prediction results of H-PINN and NLGR for *Model 3*.

training loss of H-PINN quickly converges from a significant deviation to zero. The remaining subplots in Fig. 6 display the state prediction results of H-PINN, NLGR, and DeepXDE. All three methods accurately predict the state response, with overall mean square errors of 0.0355, 2.9694×10^{-5}, and 0.6759, respectively. However, the DeepXDE exhibits a larger offset and prediction error compared to the other two approaches. Table I presents the parameter identification results for *Model 1*. The values in bold in the table indicate the estimations with the lowest error. NLGR performs best in identifying linear parameters, as they are less affected by non-linearity and dynamic coupling. DeepXDE appears to perform worse, possibly due to its simple network structure.

To better evaluate the performance of the conventional PINN, the proposed H-PINN, and NLGR, the training conditions are adjusted, and their performance indices are shown in Table II. The state prediction results of the first row of each method are displayed in Fig. 6, and their corresponding parameter identification results are provided in Table I. The number of data samples used for training impacts both training time and prediction accuracy. Increasing the number of data samples slightly slows down the training speed but results in higher precision. Additionally, even if not all system states are available for collection (e.g., only one state is obtained), the network can still perform dual-state and parameter estimation without significantly affecting the training time and mean square error of state prediction.

It can be observed that H-PINN has the fastest training time while maintaining adequate precision. On the other hand, the conventional PINN has both the longest training time and the largest prediction error. It should be noted that the discontinuity of the $sign$ function significantly impacts the performance of NLGR. However, by replacing the $sign$ function with $tanh$ and a proper constant coefficient, the training time of NLGR dramatically decreases to around 12 seconds (indicated by the numbers with underlines in Table II) while achieving almost the same MSEs as when using the $sign$ function. Additionally, applying the $sign$ function in NLGR to a more complex model can cause the identification script to get stuck. Therefore, in the identification of *Model 3*, the $tanh$ function is used for the NLGR method. The conventional PINN can hardly achiev the same prediction precision within a similar training time as the others. Increasing the network depth and width can enhance precision but would considerably increase the training time. Moreover, it would have poor state prediction performance within an acceptable time for more complex physics, so the DeepXDE approach is omitted in the followinge sections.

Fig. 7 displays the state prediction results of H-PINN and NLGR for the coupled electro-mechanical model of the cobot joint, marked as Model 3. Due to the significant impact of Coulomb friction on identification precision and speed, and with known ground truth values from simulated data, the Coulomb friction $\tau_{f,m}$ for both H-PINN and NLGR is constrained to $[4.0e-2, 5.5e-2]$ to expedite the training process. Constraints for other trainable parameters are non-negative. Additionally, the torsional angle difference $\Delta\theta$ between the motor rotor and the output shaft of the gearbox is considered to facilitate more precise identification. The top-left subplot represents the training loss of H-PINN on simulated data, while the remaining subplots showcase the state predictions of both H-PINN and NLGR. Following sufficient training, both H-PINN and NLGR can accurately predict the system state, with an overall mean square error of 0.0095 and 0.0101, respectively. However, H-PINN exhibits superior performance in handling dynamic coupling, as evidenced by the torsional angle difference plot. The predicted $\Delta\theta$ from NLGR deviates from the true value in the steady state, whereas H-PINN closely aligns better with the training data.

Table III demonstrates the parameter identification results of Model 3. It shows that H-PINN significantly surpasses NLGR in identifying non-linear and dynamically coupled terms. For instance, H-PINN achieves more accurate identification of joint stiffness compared to NLGR because the weighted MSE amplifies the importance of states with a lower order of magnitude, specifically the torsional angle difference $\Delta\theta$. This discrepancy explains the more accurate estimation of parameters for non-linear and dynamically coupled systems by H-PINN. An interesting thing to mention is that both methods estimate the Coulomb friction $\tau_{f,m}$ to the lower boundary and deviate from the truth, indicating that the discontinuity should be properly eliminated in future work.

Table IV displays the training speed and precision of H-PINN and NLGR on the more complex Model 3. Both methods allow for the inclusion of additional criteria beyond the system states, such as the torsional angle difference $\Delta\theta$, in the algorithm to enhance parameter identification accuracy. NLGR adopted the $tanh$ function to mitigate the discontinuity, resulting in a faster training speed, as the $sign$ function would cause the program to get stuck. However, H-PINN faithfully adheres to the

TABLE IV
PERFORMANCE OF H-PINN AND NLGR FOR MODEL 3 UNDER DIFFERENT TRAINING CONDITIONS

Method	Epoch	Labels	Samples	Time (s)	MSE
H-PINN	300	4	500	178.52	0.0132
	300	5	500	179.81	0.0237
	300	5+$\Delta\theta$	500	183.14	**0.0095**
NLGR ($tanh$)	–	4	500	**55.77**	**0.0130**
	–	5	500	**54.88**	**0.0121**
	–	5+$\Delta\theta$	500	**57.70**	0.0101

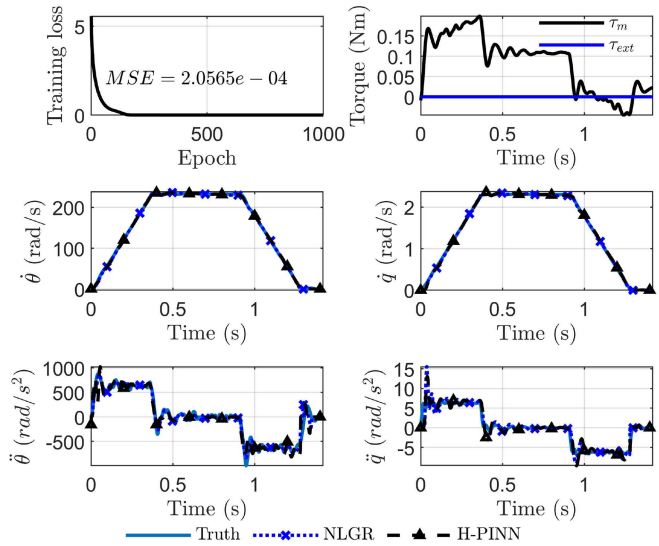

Fig. 8. Prediction results of H-PINN and NLGR for *Model 2* with experimental data.

physical model, and the training time remains acceptable even with discontinuity. Therefore, it can be concluded that H-PINN outperforms NLGR.

B. Performance of of H-PINN on Experimental Data

Since the RK4 method can only be applied to non-stiff ODEs, and experimental data may include more uncertainties than simulated data, experimental validation of H-PINN is only implemented with the less complicated Model 2 to avoid potential numerical integration problems. In the previous section, experimental data from the real robot is collected, and H-PINN with state-space Model 2 is implemented and trained with only non-negative constraints. Additionally, the NLGR method is also implemented in *MATLAB* with same initial values and constraints to provide comparison results.

The state prediction results of NLGR and H-PINN are depicted in Fig. 8. In the velocity and acceleration plots, solid lines represent the experimental data, dotted lines represent NLGR predictions, and the dashed line represents H-PINN predictions. Both methods demonstrate good alignment with the experimental system states. The final prediction errors of H-PINN and NLGR for all states are 2.0565×10^{-4} and 1.9569×10^{-4}, respectively. Their training time are 480.56 s (1000 epochs) and 638.88 s (with *sign* function), respectively. The estimated system parameters using H-PINN and NLGR are listed in Table V. Since only the reference value of the rotor inertia is known, the comparison is made only for the estimated J_m against the ground truth. The rotor inertia identification error for H-PINN is 7.85%,

TABLE V
TRAINABLE SYSTEM PARAMETERS OF MODEL 2 AND THEIR ESTIMATIONS WITH THE H-PINN AND NLGR (IN SI UNITS)

Para. Θ	Truth	H-PINN Est. ($e\%$)	NLGR Est. ($e\%$)
J_m	6e-5	**5.529e-5 (7.85)**	7.673e-5 (27.88)
J_q	–	7.780e-1	5.537e-1
c_m	–	1.018e-4	1.150e-04
c_q	–	1.799	1.688
c_1	–	18.938	18.736
k_1	–	5.164e3	5.003e3
k_2	–	9.021e8	9.003e8
$\tau_{f,m}$	–	2.291e-2	1.894e-2
$\tau_{f,q}$	–	2.296	2.678

while for NLGR, it is 27.88%. Using the estimated parameter values, the accelerations of the rotor and link are computed and compared with the experimental data. It can be observed that even though the identified parameters from H-PINN and NLGR do not coincide with each other, their state predictions with the same model still exhibit good agreement. This can be attributed to the presence of non-unique solution sets in parameter identification. Furthermore, due to the superior performance of H-PINN in handling nonlinear terms and dynamic coupling, as well as the closer estimation of rotor inertia to the dataset value, the estimation of joint dynamics from H-PINN using experimental data is considered more reliable than that of NLGR. Since the remaining five joints have the same configurations and models as the base joint, their results are omitted in this work due to the space limitations.

VII. CONCLUSION AND FUTURE WORK

In this work, the concept of PINN has been expanded to encompass complex ODE systems with nonlinear characteristics by modifying the network structure from a two-stage approach to a hybrid form (H-PINN), where the physics are integrated into the network itself, rather than being colocated alongside the network. The H-PINN has been applied to the model prediction and dynamics parameter identification of a collaborative robot joint using its state space dynamic equation. The contributions and conclusions of this work can be summarized as follows:

a) The H-PINN has been developed and implemented, embedding the physical law into the neural network through the state space dynamic equation. The RNN and customized RK4 cells improve the network's ability to handle temporal data and complex ODE systems with nonlinearity.
b) The performance of the H-PINN has been evaluated using numerically simulated data and compared with the conventional two-stage PINN and NLGR. The results highlight the H-PINN's ability to achieve high precision in model prediction and system identification, particularly when dealing with non-linearity and dynamic coupling.
c) The H-PINN has been tested using experimental data, and its results have been validated by comparing the state predictions with the ground truth. The deviation of the H-PINN's estimation of the known system parameter is much closer compared to the NLGR method, which highlights the potential of the H-PINN to be applied to real robot systems for accurate and reliable identifications.

Regarding ongoing work, the H-PINN proposed in this work is based on the RK4 method and can only solve non-stiff ODE

equations with fixed step lengths. Replacing the RK4Cell with other methods that can solve stiff ODE equations could significantly increase the robustness and application range of the proposed H-PINN.

REFERENCES

[1] L. Gualtieri, I. Palomba, E. J. Wehrle, and R. Vidoni, "The opportunities and challenges of SME manufacturing automation: Safety and ergonomics in human–robot collaboration," in Industry 4.0 for SMEs, Palgrave Macmillan, 2020, pp. 105–144.

[2] H. Yu, S. Huang, G. Chen, Y. Pan, and Z. Guo, "Human–robot interaction control of rehabilitation robots with series elastic actuators," *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1089–1100, Oct. 2015.

[3] Z. Zhou, X. Yang, H. Wang, and X. Zhang, "Coupled dynamic modeling and experimental validation of a collaborative industrial mobile manipulator with human-robot interaction," *Mechanism Mach. Theory*, vol. 176, 2022, Art. no. 105025.

[4] T. Brito, J. Queiroz, L. Piardi, L. A. Fernandes, J. Lima, and P. Leitão, "A machine learning approach for collaborative robot smart manufacturing inspection for quality control systems," *Procedia Manuf.*, vol. 51, pp. 11–18, 2020.

[5] N. D. Palo and E. Johns, "Safari: Safe and active robot imitation learning with imagination," 2020, *arXiv:2011.09586*.

[6] X. Yang, D. Qiang, Z. Chen, H. Wang, Z. Zhou, and X. Zhang, "Dynamic modeling and digital twin of a harmonic drive based collaborative robot joint," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 4862–4868.

[7] M. Taghbalout, J. F. Antoine, and G. Abba, "Experimental dynamic identification of a yumi collaborative robot," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1168–1173, 2019.

[8] G. Zhang et al., "A systematic error compensation strategy based on an optimized recurrent neural network for collaborative robot dynamics," *Appl. Sci.*, vol. 10, no. 19, 2020, Art. no. 6743.

[9] Z. Zhou, X. Yang, H. Wang, and X. Zhang, "Digital twin with integrated robot-human/environment interaction dynamics for an industrial mobile manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 5041–5047.

[10] T. J. Royston and D. Shu, "A high-fidelity harmonic drive model," *J. Dyn. Syst., Meas., Control*, vol. 134, pp. 011002–1, 2012.

[11] Y. Liu, C. Tan, Y. Zhao, H. Liu, and Y. Liu, "Nonlinear attributes modeling and analysis of harmonic drive manipulator joint," in *Proc. IEEE 3rd Int. Conf. Control, Automat. Robot.*, 2017, pp. 256–264.

[12] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, 2019.

[13] L. Ljung, "System identification," in *Signal Analysis and Prediction*, Berlin, Germany: Springer, 1998, pp. 163–173.

[14] L. Fu and P. Li, "The research survey of system identification method," in *Proc. IEEE 5th Int. Conf. Intell. Hum.- Mach. Syst. Cybern.*, 2013, pp. 397–401.

[15] L. Ljung, "Perspectives on system identification," *Annu. Rev. Control*, vol. 34, no. 1, pp. 1–12, 2010.

[16] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 2, pp. 281–304, 2019.

[17] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.

[18] G. Wang, Q.-S. Jia, J. Qiao, J. Bi, and M. Zhou, "Deep learning-based model predictive control for continuous stirred-tank reactor system," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3643–3652, Aug. 2021.

[19] J. S. Toquica, P. S. Oliveira, W. S. Souza, J. M. S. Motta, and D. L. Borges, "An analytical and a deep learning model for solving the inverse kinematic problem of an industrial parallel robot," *Comput. Ind. Eng.*, vol. 151, 2021, Art. no. 106682.

[20] E. Rueckert, M. Nakatenus, S. Tosatto, and J. Peters, "Learning inverse dynamics models in o (n) time with LSTM networks," in *Proc. IEEE-RAS 17th Int. Conf. Humanoid Robot.*, 2017, pp. 811–816.

[21] E. Madsen, O. S. Rosenlund, D. Brandt, and X. Zhang, "Model-based on-line estimation of time-varying nonlinear joint stiffness on an e-series universal robots manipulator," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 8408–8414.

[22] J.-P. Noël and J. Schoukens, "Grey-box state-space identification of nonlinear mechanical vibrations," *Int. J. Control*, vol. 91, no. 5, pp. 1118–1139, 2018.

[23] M. Raissi, Z. Wang, M. S. Triantafyllou, and G. E. Karniadakis, "Deep learning of vortex-induced vibrations," *J. Fluid Mechanics*, vol. 861, pp. 119–137, 2019.

[24] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis, "Physics-informed neural networks (PINNS) for fluid mechanics: A review," *Acta Mechanica Sinica*, vol. 37, no. 12, pp. 1–12, 2022.

[25] W. Ji, W. Qiu, Z. Shi, S. Pan, and S. Deng, "Stiff-PINN: Physics-informed neural network for stiff chemical kinetics," *J. Phys. Chem. A*, vol. 125, no. 36, pp. 8098–8106, 2021.

[26] S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk, "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture," *Theor. Appl. Fracture Mechanics*, vol. 106, 2020, Art. no. 102447.

[27] J. Nicodemus, J. Kneifl, J. Fehr, and B. Unger, "Physics-informed neural networks-based model predictive control for multi-link manipulators," *IFAC-PapersOnLine*, vol. 55, pp. 331–336, 2022.

[28] W. Sun, N. Akashi, Y. Kuniyoshi, and K. Nakajima, "Physics-informed recurrent neural networks for soft pneumatic actuators," *IEEE Robot. Automat. Lett.*, vol. 7, no. 3, pp. 6862–6869, Jul. 2022.

[29] R. G. Nascimento, K. Fricke, and F. A. Viana, "A tutorial on solving ordinary differential equations using python and hybrid physics-informed neural network," *Eng. Appl. Artif. Intell.*, vol. 96, 2020, Art. no. 103996.

[30] W. Li and K.-M. Lee, "Physics informed neural network for parameter identification and boundary force estimation of compliant and biomechanical systems," *Int. J. Intell. Robot. Appl.*, vol. 5, no. 3, pp. 313–325, 2021.

[31] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis, "DeepXDE: A deep learning library for solving differential equations," *SIAM Rev.*, vol. 63, no. 1, pp. 208–228, 2021.

[32] J. C. Wong, C. Ooi, A. Gupta, and Y.-S. Ong, "Learning in sinusoidal spaces with physics-informed neural networks," *IEEE Trans. Artif. Intell.*, early access, Jul. 19, 2022, doi: 10.1109/TAI.2022.3192362.

[33] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics-informed neural networks: Where we are and what's next," *J. Sci. Comput.*, vol. 92, 2022, Art. no. 88.

[34] J. Hollerbach, W. Khalil, and M. Gautier, "Model identification," in *Springer Handbook of Robotics*. Berlin, Germany: Springer, 2016, pp. 113–138.

[35] M. Y. Niu, L. Horesh, and I. Chuang, "Recurrent neural networks in the eye of differential equations," 2019, *arXiv:1904.12933*.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[37] E. Madsen, S. A. Timm, N. A. Ujfalusi, O. S. Rosenlund, D. Brandt, and X. Zhang, "Dynamics parametrization and calibration of flexible-joint collaborative industrial robot manipulators," *Math. Problems Eng.*, vol. 2020, pp. 1–13, 2020.