Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 54-20 (2021) 705–710

Physics-guided and Neural Network Learning-based Sliding Mode Control ⋆

Yajie Bao, Vaishnavi Thesma, and Javad Mohammadpour Velni

School of Electrical and Computer Engineering, University of Georgia, Athens, GA 30602, USA
(e-mail: {yb18054, vst16532, javadm}@uga.edu).

Abstract: This paper presents a physics-guided neural network learning fused with a sliding mode control design approach for closed-loop model identification of nonlinear systems. In particular, physical knowledge of Lagrangian for mechanical systems is incorporated into neural networks to facilitate the learning of equations of motion from data. A sliding mode controller design is considered that uses the identified model and further refines the model online by an updating law determined from the Lyapunov analysis. The proposed approach is shown to be more efficient and targeted for identification than existing Lagrangian-based neural network approaches. Simulation results on a two-joint manipulator and a three-joint manipulator demonstrate that the proposed learning-based approach can achieve better performance of model identification and improved tracking control with less computational cost compared to learning-based control without online fine-tuning.

Keywords: Physics-guided learning; Neural networks; Sliding mode control; Online learning.

1. INTRODUCTION

Machine learning methods have been increasingly employed to capture/model unknown system dynamics. However, directly applying black-box learning-based methods suffers from the challenges of model selection, data dependence, interpretability, and generalization ability (Jia et al., 2020). Leveraging existing knowledge to improve the effectiveness of data-driven modeling is gaining prominence in modeling (Karpatne et al., 2017).

Artificial neural networks (ANN) can represent a variety of functions by universal approximation theorems (e.g., Kratsios (2021)) and are widely used for modeling. However, the theorems typically do not provide a construction for the structure and weights of neural networks. Informing the design of ANN architecture by physical knowledge can assist in obtaining generalizable and scientifically interpretable results (Karpatne et al., 2017). The questions relevant to physics-informed design are *what knowledge to incorporate* and *how to embed knowledge*. The manner to embedding knowledge in the architecture affects the generalization ability and computational efficiency of modeling given a sufficient data set, which further determines the performance of the downstream work.

Lagrangian mechanics is a prominent formalism to derive the equations of motion for mechanical systems by applying calculus of variations to the Lagrangian which is a function of generalized coordinates q, capable of describing the complete dynamics of a system. Cranmer et al. (2020) proposed to use ANN to represent black-box Lagrangians (LNNs), derive numerical expressions of the Euler-Lagrange equations. LNNs embed Euler-Lagrange equations in the ANN architecture and produce energy-conserving models. However, the embedded equation is completely dependent on the Lagrangian and cannot encode the information that the inertia matrix $H(q)$ should be symmetric and positive definite. From the perspective of computational efficiency, LNNs require computing the inverse Hessian which scales $\mathcal{O}(d^3)$ where d is the number of coordinates. Moreover, pseudoinverse is used to avoid singularity, which can cause instability of LNNs training. Additionally, the learned ordinary differential equations (ODEs) are not guaranteed to have solutions, as the conditions for the existence of solutions to ODEs are not considered. Instead, Lutter et al. (2019) assume the kinetic energy $T = \dot{q}^{\mathrm{T}} H(q)\dot{q}$ and directly impose the Euler-Lagrange equations of rigid-body systems upon the ANNs structure named deep Lagrangian neural networks (DeLaN). DeLaN ensures the positive semi-definiteness of $H(q)$ and avoids the inverse of $H(q)$ by learning an inverse model. However, DeLaN cannot be directly applied to systems with other forms of Lagrangians. Additionally, Roehrl et al. (2020) use ANN to represent the unknown part in the Euler-Lagrange equations to learn a forward ODE model from data and use the reverse-mode automatic differentiation to obtain the derivatives of the loss. Furthermore, Raissi et al. (2017) proposed physics-informed deep learning to discover nonlinear partial differential equations in a given form. Generally, introducing the domain-specific knowledge can increase the generalization ability in the given domain but limits the applicability of the model in similar/related domains where such knowledge does not hold/exist. This paper focuses on the learning of forward models that can be used for model-based control and proposes a Lagrangian-informed neural network (LINN) using coarse-grained knowledge of Lagrangians without specifying the exact Lagrangian forms. Specifically, we encode the

⋆ This research was financially supported by the United States National Science Foundation under award #1762595.

10.1016/j.ifacol.2021.11.254

knowledge via feature engineering and architecture design of LINNs.

Moreover, the accuracy of physics-guided neural network models are subject to the data representativeness. Adapting the learned model online is useful to increase the accuracy of the model at current operating points and improve the performance of the model-based control (Bao et al., 2020). Considering the form of models derived from the Euler-Lagrange equations, adaptive sliding mode control (SMC), which is a well-studied technique and guarantees stability with bounded plant-model mismatch, can be used for online learning. In particular, we give the updating law of LINNs weights based on the Lyapunov stability analysis to refine the LINN model online. The objective of SMC here is not only to achieve low tracking errors but also to enhance the accuracy of the learned model.

Contribution of this paper is to propose a computationally efficient Lagrangian-informed neural networks approach for physics-guided data-driven system identification and a Lyapunov-based online learning approach to refine the model in real time.

2. PROBLEM STATEMENT AND RELATED PRELIMINARIES

Suppose that the mathematical model of a system can be expressed in the form of

$$H(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau, \tag{1}$$

where $q \in \mathbb{R}^{n_q}$ are generalized coordinates and $\tau \in \mathbb{R}^{n_\tau}$ are generalized forces. The families of functions that H, C, and G belong to are unknown but coarse-grained knowledge exists. The coarse-grained knowledge includes, but is not limited to, the basis functions of H, C, and G. For instance, when $H(q)$ is a function of $\cos(q)$, then $\cos(q)$ can be used as a feature function and the input to ANNs, which improves learning efficiency by saving the cost for ANNs to approximate the feature function from data.

The problem considered in this paper is to learn from data a forward ODE model that should be equivalent to (1) but in the form of

$$\dot{x} = \begin{bmatrix} \dot{q} \\ h(q,\dot{q},\tau) \end{bmatrix} \tag{2}$$

where $x = [q^{\mathrm{T}}, \dot{q}^{\mathrm{T}}]^{\mathrm{T}}$ using Lagrangian-informed neural networks (LINNs) and refine the model online using closed-loop data from a sliding mode controller (SMC). The robust element of SMC depends on the accuracy of the LINNs model and the updating law of LINNs weights is derived based on the Lyapunov stability analysis of SMC.

2.1 Preliminaries on Lagrangian Mechanics

Lagrangian mechanics is widely used to solve mechanical problems in physics. The central quantity of Lagrangian mechanics is the Lagrangian L, which is a function of q, $\dot{q}$, and time t. An L is valid if it generates correct dynamics of the entire system and follows physical laws. There is no unified expression of L for all physical systems. The $L = T - V$ in Lutter et al. (2019) can be used for non-relativistic Lagrangian mechanics but does not hold in relativistic Lagrangian mechanics.

Using the calculus of variations, we have the Euler-Lagrange equations as

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial \dot{q}_i}\right) = \frac{\partial L}{\partial q_i}. \tag{3}$$

The equations of motion of the system can be derived by substituting L into (3) and add the generalized forces to the right-hand side, which results in (1). These equations bypass constraint forces and are ODEs in spite of the partial derivatives.

2.2 Sliding Mode Control

Sliding model control (SMC) is a variable structure control method that has been widely used for a broad class of nonlinear systems (Bartoszewicz and Żuk (2010)). We consider the second-order system analyzed in (Liu and Wang, 2011), as an example, in the form of

$$\ddot{\theta} = f(\theta,\dot{\theta}) + g(\theta,\dot{\theta})u + d(t), \tag{4}$$

where θ, u, and $d(t)$ are the output, control input, and bounded external disturbance, respectively, and $|d(t)| \leq D$. It is noted that (1) can be transformed into (4) with $f = -H^{-1}(C+G)$ and $g = H^{-1}$. Using the Lyapunov function candidate $L_{\mathrm{y}} = \frac{1}{2}s^2$, SMC applies discontinuous control signal

$$u = g^{-1}(-f + \ddot{\theta}_{\mathrm{d}} + k\dot{e} + \eta_0 \mathrm{sgn}(s)), \tag{5}$$

where $k > 0$ is a design parameter of the selected surface $s = \dot{e} + ke$, $e = \theta - \theta_{\mathrm{d}}$ denotes the tracking error with respect to the desired output θ_{d}, and $\eta_0 \geq D$ are feedback gains dominating $d(t)$ such that system trajectories reach the sliding surface s in finite time and then slide on this surface to the origin. The advantage of SMC lies in its robustness to the parameter variations of f and g. The inexact knowledge of f and g (i.e., the plant-model mismatch from data-driven system identification) can be compensated for by $\eta(\theta) + \eta_0$, which motivates us to use SMC to collect closed-loop data.

However, the delays in switching result in "chattering" around the sliding surface in practice (Utkin and Hoon Lee, 2006). The degree of chattering depends on the magnitude of $|\eta(\theta) + \eta_0|$, which provides the information of the accuracy of the learned physics-guided model. Adaptive SMC adapts the estimation of f online to achieve better control performance. Instead, we use adaptive SMC for online learning of LINNs in Section 3.2.

3. LAGRANGIAN-INFORMED NEURAL NETWORKS AND LYAPUNOV-BASED ONLINE LEARNING

This section introduces the techniques employed by LINNs for knowledge incorporation and online adaptation.

3.1 Lagrangian-informed Neural Networks

Feature engineering for knowledge embedding: In machine learning, a feature is an individual measurable property or characteristic of a phenomenon being observed (Bishop, 2006), and choosing features is crucial for effective algorithms. Deep learning reduces the efforts involved in extracting features (aka feature engineering) from raw data by designing efficient architecture and training on large amounts of data, which decreases the interpretability but increases the complexity of ANNs. A neural network with one 20-unit hidden layer was used to approximate the sine function. Using domain knowledge to extract features can facilitate learning efficiency. A typical coarse-grained knowledge is that sine and cosine functions of the

generalized coordinates exist in the Lagrangians and the arguments of H, C and G. We can augment $q, \dot{q}$ with $\cos(q), \sin(q)$ and designate the inputs to ANNs.

Architecture design: The information that H is symmetric and positive definite can be encoded into the architecture of ANNs. In particular, $\hat{H}$ can be modeled as Lutter et al. (2019) by $\hat{H} = L_M L_M^{\mathrm{T}}$, where L_M is a lower triangular matrix with positive diagonal entries and represented by a neural network. However, the representation of the vectorized $\hat{H}$

$$\hat{\boldsymbol{H}} = (\phi_H \hat{W}_H)^{\mathrm{T}} \tag{6}$$

where ϕ_H is the basis function and $\hat{W}_H$ is the weight matrix, is more suitable for online learning as discussed in Section 3.2. To have (6) and ensure $\hat{H} = \hat{H}^{\mathrm{T}} \succ 0$, we propose to use a convex combination of positive definite matrices $\hat{H}_v = L_{M,v} L_{M,v}^{\mathrm{T}}$ to represent $\hat{H}$, i.e.,

$$\hat{H} = \sum_{v=1}^{V} \alpha_v \hat{H}_v = \sum_{v=1}^{V} \alpha_v L_{M,v} L_{M,v}^{\mathrm{T}}, \tag{7}$$

$$\alpha_v \geq 0, \sum_{v=1}^{V} \alpha_v = 1, \tag{8}$$

as the convex combination of positive definite matrices is still positive definite. Furthermore, the expression of $C(q, \dot{q})$ in Lutter et al. (2019) is derived from the specific kinetic energy, which decreases the accuracy of learned model when T is not in the specified form. To enable LINNs to be applicable to arbitrary forms of Lagrangians, we directly use networks with physics-guided features as inputs to represent C and G, which also simplifies the complexity of the computation graph of LINNs.

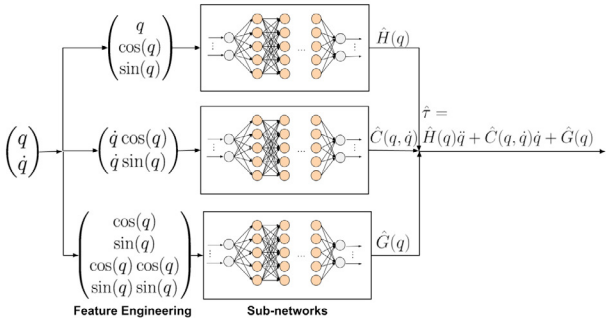


Fig. 1. The architecture of LINNs.

The architecture of LINNs is shown in Fig. 1. This architecture significantly improves the computational efficiency, compared with Cranmer et al. (2020) where explicit inverse of the H derived from parameterized Lagrangians is implemented. The loss function of LINNs is the mean squared error (MSE)

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} (\tau - \hat{\tau})^2 \tag{9}$$

where N is the number of data points. This formulation views the data as i.i.d. sampled without considering the error dynamics, which explains the large errors of multi-step predictions and necessitates the online learning using closed-loop data. The forward ODE model (2) can be estimated using

$$\hat{h}(q, \dot{q}, \tau) = \hat{H}^{-1}(q)(\tau - \hat{C}(q, \dot{q}) - \hat{G}(q)). \tag{10}$$

Solutions to The ODE model: By the Picard–Lindelöf theorem, if $\hat{h}$ is uniformly Lipschitz continuous in $(q, \dot{q}, \tau)$ and continuous in t, then for some value $\epsilon > 0$, there exists a unique solution $q(t)$ to the initial value problem on the interval $[t_0 - \epsilon, t_0 + \epsilon]$. Most activation functions have a Lipschitz constant that is one (1) such as Leaky ReLU, SoftPlus, and Tanh. Additionally, the Lipschitz constants of deep neural networks can be estimated using SeqLip algorithm proposed in Scaman and Virmaux (2018). Therefore, $\hat{h}$ is uniformly Lipschitz continuous on the admissible set of $(q, \dot{q}, \tau)$.

Generalization of LINNs: The accuracy of LINNs is evaluated by the generalization, i.e., that the population (or testing) error is close to the training error by minimizing the training error (Neyshabur et al., 2017). However, the training error is defined as (9) instead of the best fit ratio for model accuracy $\mathrm{BFR} = 100\% \cdot \max\left(1 - \frac{\|x - \hat{x}\|_2}{\|x - \bar{x}\|_2}, 0\right)$ where $\hat{x}$ is the solution to (2) and $\bar{x}$ is the mean of the real trajectory x, which suggests LINNs with small testing errors are not necessarily a good ODE model while a good ODE model must have a small testing error with respect to τ. Therefore, fine-tuning LINNs is necessary. However, using the reverse-mode differentiation of an ODE solution Chen et al. (2018) to directly tune the ODE model suffers from the low computational efficiency. Instead, we propose the following approach to fine-tune LINNs online while achieving superb tracking control performance.

3.2 Lyapunov-based Online Learning

We first show the Lyapunov stability analysis of SMC. The sliding variable considered is $s = \dot{e} + \Lambda e$ where $\Lambda = \Lambda^{\mathrm{T}} > 0$ and $e = q_{\mathrm{d}}(t) - q(t)$ is the tracking error and the Lyapunov function is $L_{\mathrm{y}} = \frac{1}{2} s^{\mathrm{T}} H s$. Then

$$\dot{L}_{\mathrm{y}} = s^{\mathrm{T}} H \dot{s} + \frac{1}{2} s^{\mathrm{T}} \dot{H} s. \tag{11}$$

$$\begin{aligned} H\dot{s} &= H(\ddot{e} + \Lambda \dot{e}) = H(\ddot{q}_{\mathrm{d}} + \Lambda \dot{e}) - H\ddot{q} \\ &= H(\ddot{q}_{\mathrm{d}} + \Lambda \dot{e}) + C\dot{q} + G - \tau \\ &= H(\ddot{q}_{\mathrm{d}} + \Lambda \dot{e}) + C(\dot{q}_{\mathrm{d}} - s + \Lambda e) + G - \tau \\ &= -Cs - \tau + p \end{aligned} \tag{12}$$

where $p = H(\ddot{q}_{\mathrm{d}} + \Lambda \dot{e}) + C(\dot{q}_{\mathrm{d}} + \Lambda e) + G$. Using the controller $\tau = p + K_{\mathrm{v}} s$ and the skew symmetry property $\dot{H} - 2C = 0$, we have

$$\dot{L}_{\mathrm{y}} = -s^{\mathrm{T}} K_{\mathrm{v}} s + \frac{1}{2} s^{\mathrm{T}} (\dot{H} - 2C) s = -s^{\mathrm{T}} K_{\mathrm{v}} s \leq 0, \tag{13}$$

where K_{v} is a symmetric positive definite constant matrix. However, calculation of p involves H, C and G that are learned from data. The mismatch $\epsilon = p - \hat{p}$ between the ideal p and the estimated $\hat{p}$ changes (13) to

$$\dot{L}_{\mathrm{y}} = -s^{\mathrm{T}} K_{\mathrm{v}} s + s^{\mathrm{T}} \epsilon \tag{14}$$

using $\tau = \hat{p} + K_{\mathrm{v}} s$, which can cause the violation of $\dot{L}_{\mathrm{y}} \leq 0$. To compensate for the mismatch, a robust term v is added to the control input and the control inputs are given by

$$\tau = \hat{p} + K_{\mathrm{v}} s - v. \tag{15}$$

Using the control law (15), we obtain

$$\dot{L}_{\mathrm{y}} = -s^{\mathrm{T}} K_{\mathrm{v}} s + s^{\mathrm{T}} (\epsilon + v) \leq 0 \tag{16}$$

if $v = -\epsilon_N \mathrm{sgn}(s)$ and $|\epsilon| < \epsilon_N$. Additionally, a continuous function

$$\mathrm{sat}(s) = \begin{cases} 1 - e^{-s/\gamma}, s \geq 0 \\ -(1 - e^{s/\gamma}), s < 0 \end{cases}$$

where $\gamma > 0$ is a small design parameter can be used to replace $\mathrm{sgn}(s)$, which reduces chattering.

Moreover, when p is approximated by $\hat{p} = \hat{W}^{\mathrm{T}}\phi(\mathrm{x})$ and $\hat{W}$ are tuned by $\dot{\hat{W}} = F_W \phi s^{\mathrm{T}}$ where

$$\mathrm{x} = \left[e^{\mathrm{T}}\ \dot{e}^{\mathrm{T}}\ q_{\mathrm{d}}^{\mathrm{T}}\ \dot{q}_{\mathrm{d}}^{\mathrm{T}}\ \ddot{q}_{\mathrm{d}}^{\mathrm{T}}\right]^{\mathrm{T}},$$

$\phi(\mathrm{x})$ is a suitable basis, and $F_W = F_W^{\mathrm{T}} \succ 0$ is a constant matrix, then the control law (15) achieves uniformly ultimately bounded (UUB) filtered tracking error $s(t)$ and $\hat{W}$ is bounded by Theorem 3.5 in (Lewis et al., 1995) under the conditions that the desired trajectories $q_{\mathrm{d}}, \dot{q}_{\mathrm{d}}, \ddot{q}_{\mathrm{d}}$ are bounded and $\epsilon_N(x) = \alpha_0(q_{\mathrm{d}}, \dot{q}_{\mathrm{d}}, \ddot{q}_{\mathrm{d}}) + \rho(s)$ where α_0 and ρ are continuous polynomial functions. Furthermore, the tracking error can be made as small as possible by choosing large control gain K_{v}.

When using ANNs to represent $\hat{H}, \hat{C}, \hat{G}$, fine-tuning weights $\hat{W}_H, \hat{W}_C, \hat{W}_G$ in the last layer of ANNs can increase the computational efficiency of online learning. Furthermore, with a small testing error, it is reasonable to assume the hidden layers provide suitable basis $\phi_H^{\mathrm{T}} \in \mathbb{R}^{d_H}, \phi_C^{\mathrm{T}} \in \mathbb{R}^{d_C}, \phi_G^{\mathrm{T}} \in \mathbb{R}^{d_G}$ [1] and the vectorized matrices are calculated by $\hat{\boldsymbol{H}} = (\phi_H \hat{W}_H)^{\mathrm{T}}, \hat{\boldsymbol{C}} = (\phi_C \hat{W}_C)^{\mathrm{T}}, \hat{\boldsymbol{G}} = (\phi_G \hat{W}_G)^{\mathrm{T}}$. By reshaping $\hat{W}_i^{\mathrm{T}} \in \mathbb{R}^{n_q^2 \times d_i}$ into $\hat{\mathrm{W}}_i \in \mathbb{R}^{n_q \times d_i \times n_q}$, $i = H, C$ and $\hat{\mathrm{W}}_G = \hat{W}_G^{\mathrm{T}} \in \mathbb{R}^{n_q \times d_G}$, then

$$\hat{p}(x) = \begin{bmatrix}\hat{\mathrm{W}}_H\ \hat{\mathrm{W}}_C\ \hat{\mathrm{W}}_G\end{bmatrix} \begin{bmatrix}\varsigma_1(t) \otimes \phi_H^{\mathrm{T}} \\ \varsigma_2(t) \otimes \phi_C^{\mathrm{T}} \\ \phi_G\end{bmatrix} := \hat{W}^{\mathrm{T}} \begin{bmatrix}\Phi_H \\ \Phi_C \\ \Phi_G\end{bmatrix} \quad (17)$$

where $\otimes$ denotes the Kronecker product, $\varsigma_1(t) = \ddot{q}_{\mathrm{d}} + \Lambda\dot{e}$, $\varsigma_2(t) = \dot{q}_{\mathrm{d}} + \Lambda e$ and the weight update law is

$$\begin{aligned}\dot{\hat{\mathrm{W}}}_H &= F_H \Phi_H s^{\mathrm{T}} - \kappa_H F_H \|s\| \hat{\mathrm{W}}_H \\ \dot{\hat{\mathrm{W}}}_C &= F_C \Phi_C s^{\mathrm{T}} - \kappa_C F_C \|s\| \hat{\mathrm{W}}_C \\ \dot{\hat{\mathrm{W}}}_G &= F_G \Phi_G s^{\mathrm{T}} - \kappa_G F_G \|s\| \hat{\mathrm{W}}_G\end{aligned} \quad (18)$$

where $-\kappa_i F_i \|s\| \hat{\mathrm{W}}_i$ (aka e-modification in (Narendra and Annaswamy, 1987)) are used to relax the persistence of excitation conditions on ϕ_i, and $F_i, \kappa_i \succ 0$, $i = H, C, G$, are design parameters in the form of $n_q \times n_q$ block diagonal matrices. Moreover, when using (6) to represent H, the number of weights that need fine-tuning after fixing $\hat{H}_v$'s is reduced from $n_q^2 \times d_h$ in (18) to the number V of $\hat{H}_i$'s, which further improves the online fine-tuning efficiency at the expense of increasing the offline learning complexity of LINNs. Additionally, the updating law is simplified to

$$\begin{aligned}\dot{\hat{\mathrm{W}}}_H =& F_H[\varsigma_1(t)^{\mathrm{T}} \hat{H}_1 s, \cdots, \varsigma_1(t)^{\mathrm{T}} \hat{H}_V s]^{\mathrm{T}} - \\ & \kappa_H F_H \|s\| \hat{\mathrm{W}}_H\end{aligned} \quad (19)$$

where $F_H, \kappa_H \succ 0$ are $V \times V$ diagonal matrices. Moreover, we apply softmax transformation to the updated $\hat{\mathrm{W}}$ such that the combination of $\hat{H}_i$ remains convex. The schematic of online fine-tuning LINNs is shown in Fig. 2. At each time step, the control input (15) is estimated using the parameters updated by (18).

4. EXPERIMENTAL RESULTS AND VALIDATION

This section validates the proposed learning methods.

[1] We use d_i to denote the dimension of ϕ_i, $i = H, C, G$.

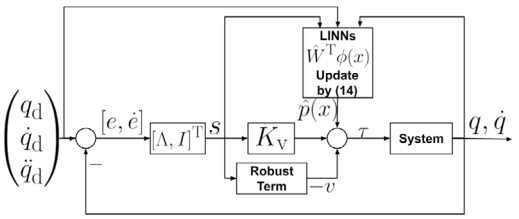


Fig. 2. Schematic of online learning of LINNs.

4.1 Validation on A 2-joint Manipulator

The kinetic equation of the 2-joint manipulator adapted from (Liu and Wang, 2011) is in the form of (1) and

$$\begin{aligned}H(q) &= \begin{bmatrix}\alpha_1 + \alpha_2 + 2\alpha_3\cos(q_2) & \alpha_2 + \alpha_3\cos(q_2) \\ \alpha_2 + \alpha_3\cos(q_2) & \alpha_2\end{bmatrix}, \\ C(q,\dot{q}) &= \begin{bmatrix}-\alpha_3\dot{q}_2\sin(q_2) & -\alpha_3(\dot{q}_1 + \dot{q}_2)\sin(q_2) \\ \alpha_3\dot{q}_1\sin(q_2) & 0\end{bmatrix}, \\ G(q) &= \begin{bmatrix}\alpha_4 g\cos(q_1) + \alpha_5 g\cos(q_1 + q_2) \\ \alpha_5 g\cos(q_1 + q_2)\end{bmatrix}, \\ \tau &= \tau_u - F(\dot{q}) = \tau_u - 0.02\mathrm{sgn}(\dot{q}),\end{aligned} \quad (20)$$

where $[\alpha_i]_{i=1}^5 = [2.9\ 0.76\ 0.87\ 3.04\ 0.87]$, τ_u is the control input, and $F(\dot{q})$ is the friction force.

Experimental setup: A pseudo-random binary sequence (PRBS) is used to excite the model with the amplitude $\tau_1 = \{30, 50\}, \tau_2 = \{5, 15\}$ and frequency in the range of $[0.1, 10]$. A sampling time of 0.01 *sec* is chosen and $5,000$ data points were collected. We used the first $4,000$ points for training and the remainder for testing.

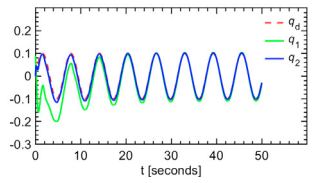
Based on the properties of the dynamics of two-joint manipulator, the feature functions are selected as

$$\begin{aligned}\phi_H &= \left[q^{\mathrm{T}}\ \cos(q)^{\mathrm{T}}\ \sin(q)^{\mathrm{T}}\right], \\ \phi_C &= \left[[\dot{q}\cos(q)]^{\mathrm{T}}\ [\dot{q}\sin(q)]^{\mathrm{T}}\right], \\ \phi_G &= \left[\cos(q)^{\mathrm{T}}\ \sin(q)^{\mathrm{T}}\ \cos^2(q)^{\mathrm{T}}\ \sin^2(q)^{\mathrm{T}}\right], \\ \phi_F &= \left[\dot{q}^{\mathrm{T}}\ \mathrm{sgn}(\dot{q})^{\mathrm{T}}\right].\end{aligned} \quad (21)$$

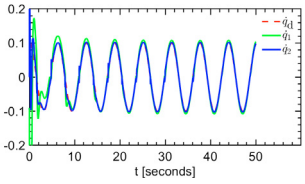
Then, the features are respectively fed into the input layers of the subnetworks in LINNs to represent H, C, G and F. In particular, considering the model complexity of manipulators, the input layers are directly connected to the output layers without activation functions for all the subnetworks. The output vectors of subnetworks are reshaped into matrices which are used to represent (10). The trained LINNs are fine-tuned in real time using the approach in Section 3.2. The initial state $x_0 = [0.09\ 0\ -0.09\ 0]^{\mathrm{T}}$ and the reference signal $q_{\mathrm{d}} = 0.1\sin(t)$ for both joints were considered. The controller parameters were assumed to be $\Lambda = \mathrm{diag}\{5, 5\}$ $K_{\mathrm{v}} = \mathrm{diag}\{20, 20\}$. Moreover, the robust term in (15) is not used when online learning is active to avoid chattering. Finally, $F_i = I_i$, $i = H, C, G, F$ and $\kappa_i = 0.1 I_i$ are used for (18) where I_i's are identity matrices with suitable dimensions.

Results and discussion: The upper bound of the mismatch is determined to be $\epsilon_{\max} = p - \hat{p} = [40.41\ 5.82]$. $\epsilon_N = \epsilon_{\max}$ can result in significant chattering while smaller ϵ_N leads to larger tracking errors. The MSE on the training data set is 1187.87. Moreover, the learned ODE model cannot be solved at some states and inputs using the solver

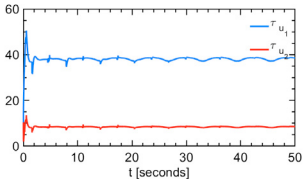
that works for the plant model. Fig. 3 shows the tracking results and control inputs and Fig. 4(a) shows one-step predictions of the fine-tuned LINNs. As observed, both the tracking and prediction errors converge to 0 after a few steps of online learning.



(a) Position tracking.

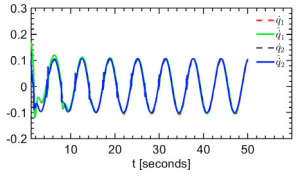


(b) Velocity tracking.

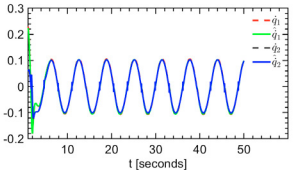


(c) Control inputs.

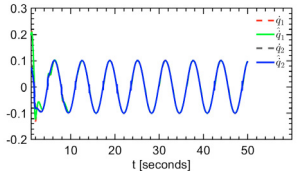
Fig. 3. SMC results for the 2-joint manipulator.



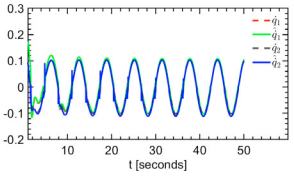
(a) $\text{BFR}_{\dot{q}} = [95.49\%\ 89.27\%]^{\text{T}}$ using linear combinations of features in (21).



(b) $\text{BFR}_{\dot{q}} = [96.30\%\ 97.66\%]^{\text{T}}$ using 6-layer ANNs with features in (21) as input.



(c) $\text{BFR}_{\dot{q}} = [98.19\%\ 94.59\%]^{\text{T}}$ using linear combination of augmented features.



(d) $\text{BFR}_{\dot{q}} = [95.22\%\ 88.86\%]^{\text{T}}$ using LINNs architecture in (6).

Fig. 4. Simulated $\hat{\dot{q}}$ of the fine-tuned LINNs model and $\dot{q}$ of the original system.

It is noted that the LINNs architecture in the experimental setup cannot represent the family of models to which (20) belongs, as (21) does not contain such terms as $\cos(q_1 + q_2)$ which is not known a priori. Therefore, we used multilayer ANNs to compensate for insufficiency of features. Specifically, we used 4 hidden layers with 64 units and tanh activation. The tuning parameters of using multilayer ANNs are $\Lambda = \text{diag}\{15, 15\}$, $K_{\text{v}} = \text{diag}\{5, 5\}$, $\epsilon_N = [0.001; 0.001]$, $F_i = 2I_i, i = H, C, G, F$ and $\kappa_i = 1e-4I_i$ and the results are shown in Fig. 4(b). Additionally, we augmented ϕ_C with $\dot{q} \otimes \cos(q), \dot{q} \otimes \sin(q)$ and ϕ_G with $\cos(q) \otimes \cos(q), \sin(q) \otimes \sin(q)$ in (21) such that linear combination of the augmented features can represent (20). The tuning parameters of using the augmented features are the same as the experimental setup except that $F_i = 10I_i$ and $\kappa_i = 0.1I_i$, and the results are shown in Fig. 4(c). The BFRs are not 100% even though the LINNs with augmented features contain the families of functions where H, C, G, F belong, which demonstrates the effects of data and optimization on the model accuracy. Moreover, LINNs using 6-layer ANNs achieve similar accuracy to LINNs using the augmented features, which shows the ANNs can be used to learn a good model with coarse-grained knowledge.

Moreover, we tested the performance of LINNs using the architecture (6) and updating law (19). In particular, V=4. A 3-layer ANN with leaky ReLU activation is used as the base network. Then, a layer with linear activation is added to the base network to model the off-diagonal elements $L^o_{M,v}$ and another layer with softplus activation is added to the base network to model the diagonal elements $L^d_{M,v}$ of $L_{M,v}$. Moreover, 4-layer ANNs with (21) as inputs and tanh as activation functions are used to represent C and G. The tuning parameters for LINNs with the architecture (6) are the same as the experimental setup except that $K_{\text{v}} = \text{diag}\{25, 25\}$, $F_i = I_i$ and $\kappa_i = I_i$. As shown in Fig. 4(d), using the architecture (6) improves online learning efficiency by fine-tuning less parameters but still achieves an accuracy comparable with LINNs using multilayer ANNs. Additionally, increasing the number V of $\hat{H}_v$'s can improve the accuracy of fine-tuned LINNs model.

4.2 Validation on A 3-joint Manipulator

We further validated the proposed approach on a 3-joint manipulator. The motion equations of the 3-joint manipulator are borrowed from (Truong et al., 2019). The highest degree terms in H, C, G are $\sin^2(q_2 + q_3), \sin(q_2 + q_3)\cos(q_2 + q_3)\dot{q}_1, \sin(q_2 + q_3)$, respectively.

Experimental setting: First, we assume that a dictionary of feature functions exists such that H, C, G are affine functions of the features. Then, the LINNs share the same architecture as 4.1.1. Additionally, there are 5 closed-loop trajectories generated by tracking results and

$$\begin{aligned} q_{1\text{d}} &= \frac{55}{2} + \frac{55}{2}\sin(2\pi t/T_i - \pi/2), \\ q_{2\text{d}} &= \frac{85}{2} + \frac{85}{2}\sin(2\pi t/T_i - \pi/2), \\ q_{3\text{d}} &= 50 + 47\sin(2\pi t/T_i - \pi/2) \end{aligned}$$

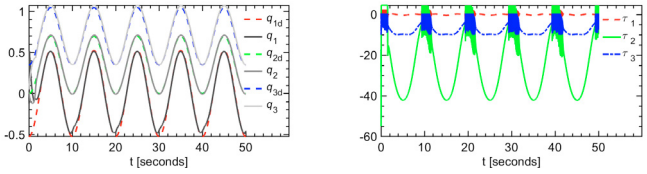
where $\{T_i\}_{i=1}^5 = \{5, 7, 10, 13, 15\}$. The sampling time is 0.01 s. One period of data per trajectory (i.e., 5,000 data points) are used to learn LINNs. Then, the trained LINNs are fine-tuned in real time using the approach in Section 3.2. The initial state $x_0 = [0.0001\ 0.01\ 0.05\ 0\ 0\ 0]^{\text{T}}$ and the reference signal

$$\begin{aligned} q_{1\text{d}} &= 30 + 30\sin(2\pi t/10 - \pi/2), \\ q_{2\text{d}} &= 20 + 20\sin(2\pi t/10 - \pi/2), \\ q_{3\text{d}} &= 40 + 20\sin(2\pi t/10 - \pi/2). \end{aligned}$$

The controller parameters are assumed $\Lambda = \text{diag}\{5, 15, 15\}$, $K_{\text{v}} = \text{diag}\{5, 60, 5\}$, and $\epsilon_N = [0.01\ 0.01\ 0.001]^{\text{T}}$. It is noted that ϵ is far less than the plant-model mismatch but the controller still works, as online learning keeps reducing the mismatch. $F_i = 0.01I_i, i = H, C, G, F$ and $\kappa_i = 0.01I_i$ are used for (18) where I_i's are identity matrices with suitable shapes.

Results and discussion: The tracking results and control inputs are shown in Fig. 5. The large variations of control

inputs result from the large K_v which is related to the reaching time required to approach the switching manifolds and used to dominate the plant-model mismatch. The identification results are shown in Fig. 6. The prediction of $\hat{\dot{q}}_1$ is not accurate even after online learning. The reason for the low BFR is the large MSE (66.83) of the initially learned LINNs model determined by the insufficient training data. However, the adaptive SMC still achieved acceptable control results and improved the identification results while the initially learned model cannot be directly used for prediction. Furthermore, using the closed-loop data to fine-tune the LINNs offline can improve the control and identification results.



(a) Position tracking. (b) Control inputs.

Fig. 5. SMC results of 3-joint manipulator.

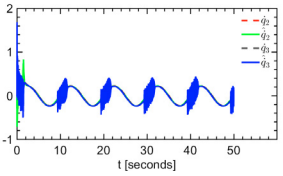


Fig. 6. Simulated $\hat{\dot{q}}_{2,3}$ of the fine-tuned LINNs model and $\dot{q}_{2,3}$ of the original system. $\text{BFR}_{\dot{q}} = [91.40\%\ 89.40\%]^{\text{T}}$.

5. CONCLUDING REMARKS

In this paper, a physics-guided neural network learning approach associated with online fine-tuning was proposed to learn an accurate model from data. The proposed network architecture (LINNs) is computationally efficient for modeling systems that can be described by Euler-Lagrange equations and compatible for SMC design, which is robust to the mismatches between the learned model and the plant. Moreover, the learned LINNs are fine-tuned in real time by an updating law derived using the Lyapunov analysis of SMC. Experiments on a 2-joint manipulator and a 3-joint manipulator models showed that the proposed methods can learn an accurate model and achieve good control performance with bounded plant-model mismatch and be applied to systems of high complexities.

REFERENCES

Bao, Y., Mohammadpour Velni, J., and Shahbakhti, M. (2020). An online transfer learning approach for identification and predictive control design with application to rcci engines. In *Dynamic Systems and Control Conference*, volume 84270, V001T21A003. American Society of Mechanical Engineers.

Bartoszewicz, A. and Żuk, J. (2010). Sliding mode control — basic concepts and current trends. In *2010 IEEE International Symposium on Industrial Electronics*, 3772–3777. doi:10.1109/ISIE.2010.5637990.

Bishop, C.M. (2006). *Pattern recognition and machine learning*. springer.

Chen, R.T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. (2018). Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6572–6583.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. (2020). Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*.

Jia, X., Willard, J., Karpatne, A., Read, J.S., Zwart, J.A., Steinbach, M., and Kumar, V. (2020). Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles. *arXiv preprint arXiv:2001.11086*.

Karpatne, A., Atluri, G., Faghmous, J.H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V. (2017). Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10), 2318–2331. doi:10.1109/TKDE.2017.2720168.

Kratsios, A. (2021). The universal approximation property. *Annals of Mathematics and Artificial Intelligence*, 1–35.

Lewis, F.L., Liu, K., and Yesildirek, A. (1995). Neural net robot controller with guaranteed tracking performance. *IEEE Transactions on Neural Networks*, 6(3), 703–715.

Liu, J. and Wang, X. (2011). *Neural Network Sliding Mode Control*, 281–300. Springer Berlin Heidelberg, Berlin, Heidelberg. doi:10.1007/978-3-642-20907-9_10.

Lutter, M., Ritter, C., and Peters, J. (2019). Deep Lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*.

Narendra, K. and Annaswamy, A. (1987). A new adaptive law for robust adaptation without persistent excitation. *IEEE Transactions on Automatic control*, 32(2), 134–145.

Neyshabur, B., Bhojanapalli, S., McAllester, D., and Srebro, N. (2017). Exploring generalization in deep learning. *Neural Information Processing Systems (NeurIPS)*.

Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2017). Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations. *CoRR*, abs/1711.10566. URL `http://arxiv.org/abs/1711.10566`.

Roehrl, M.A., Runkler, T.A., Brandtstetter, V., Tokic, M., and Obermayer, S. (2020). Modeling system dynamics with physics-informed neural networks based on lagrangian mechanics. *arXiv preprint arXiv:2005.14617*.

Scaman, K. and Virmaux, A. (2018). Lipschitz regularity of deep neural networks: analysis and efficient estimation. *arXiv preprint arXiv:1805.10965*.

Truong, H.V.A., Tran, D.T., Ahn, K.K., et al. (2019). A neural network based sliding mode control for tracking performance with parameters variation of a 3-dof manipulator. *Applied Sciences*, 9(10), 2023.

Utkin, V. and Hoon Lee (2006). Chattering problem in sliding mode control systems. In *International Workshop on Variable Structure Systems, 2006. VSS'06.*, 346–350. doi:10.1109/VSS.2006.1644542.