Phy-Taylor: Partially Physics-Knowledge-Enhanced Deep Neural Networks via NN Editing

Yanbing Mao, Yuliang Gu, Lui Sha, *Life Fellow, IEEE*, Huajie Shao, *Member, IEEE*, Qixin Wang, and Tarek Abdelzaher, *Fellow, IEEE*

***Abstract*— Purely data-driven deep neural networks (DNNs) applied to physical engineering systems can infer relations that violate physics laws, thus leading to unexpected consequences. To address this challenge, we propose a physics-knowledge-enhanced DNN framework called Phy-Taylor, accelerating learning-compliant representations with physics knowledge. The Phy-Taylor framework makes two key contributions; it introduces a new architectural physics-compatible neural network (PhN) and features a novel compliance mechanism, which we call *physics-guided neural network (NN) editing*. The PhN aims to directly capture nonlinear physical quantities, such as kinetic energy, electrical power, and aerodynamic drag force. To do so, the PhN augments NN layers with two key components: 1) monomials of the Taylor series for capturing physical quantities and 2) a suppressor for mitigating the influence of noise. The NN editing mechanism further modifies network links and activation functions consistently with physics knowledge. As an extension, we also propose a self-correcting Phy-Taylor framework for safety-critical control of autonomous systems, which introduces two additional capabilities: 1) safety relationship learning and 2) automatic output correction when safety violations occur. Through experiments, we show that Phy-Taylor features considerably fewer parameters and a remarkably accelerated training process while offering enhanced model robustness and accuracy.**

***Index Terms*— Knowledge compliance, neural network (NN) editing, physics-compatible NN (PhN).**

Nomenclature

$\mathbb{R}^n$	Set of n-dimensional real vectors.
$\mathbb{R}_{\geq 0}$	Set of nonnegative real numbers.
$\mathbb{N}$	Set of natural numbers.
$[\mathbf{x}]_i$	ith entry of vector $\mathbf{x}$.
$[\mathbf{x}]_{i:j}$	Sub-vector formed by the ith to jth entries of vector $\mathbf{x}$.
$[\mathbf{W}]_{i,j}$	Element at row i and column j of matrix $\mathbf{W}$.
$[\mathbf{W}]_{i,:}$	ith row of matrix $\mathbf{W}$.
$[\mathbf{x}\ ;\ \mathbf{y}]$	Stacked (tall column) vector of vectors $\mathbf{x}$ and $\mathbf{y}$.
$\mathbf{0}_n$	n-dimensional vector of all zeros.
$\mathbf{1}_n$	n-dimensional vector of all ones.
$\mathbf{O}_{m\times n}$	$m \times n$-dimensional zero matrix.
$\mathbf{I}_n$	$n \times n$-dimensional identity matrix.
$\|\cdot\|$	Euclidean norm of a vector or absolute value of a number.
$\odot$	Hadamard product.
$\bullet$	Multiplication operator.
$\text{len}(\mathbf{x})$	Length of vector $\mathbf{x}$.
act	Activation function.
sus	Suppressor function.

Manuscript received 27 November 2022; revised 19 June 2023; accepted 15 October 2023. This work was supported by the National Science Foundation (NSF) under Grant CPS-2311084, Grant CPS-2311085, and Grant CPS-2311086. *(Corresponding author: Yanbing Mao.)*

Yanbing Mao is with the Engineering Technology Division, Wayne State University, Detroit, MI 48201 USA (e-mail: hm9062@wayne.edu).

Yuliang Gu is with the Department of Mechanical Engineering, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: yuliang3@illinois.edu).

Lui Sha and Tarek Abdelzaher are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: lrs@illinois.edu; zaher@illinois.edu).

Huajie Shao is with the Department of Computer Science, College of William and Mary, Williamsburg, VA 23185 USA (e-mail: hshao@wm.edu).

Qixin Wang is with the Department of Computing, Hong Kong Polytechnic University, Hong Kong, SAR, China (e-mail: qixin.wang@polyu.edu.hk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2023.3325432.

Digital Object Identifier 10.1109/TNNLS.2023.3325432

I. Introduction

THE article proposes a novel physics-knowledge-enhanced deep neural network (DNN) framework called Phy-Taylor that addresses a critical flaw in purely data-driven neural networks (NNs) when used to model aspects of physical engineering systems. Namely, it addresses the potential lack of agreement between learned latent NN representations and prior physics knowledge: a flaw that sometimes leads to catastrophic consequences [1]. As visualized in Fig. 1, the Phy-Taylor framework introduces two contributions: the physics-compatible NNs (PhNs) and the physics-guided NN editing mechanism, aiming at ensuring compliance with prior physics knowledge.

The work contributes to emerging research on physics-enhanced NNs. Current approaches include physics-informed NNs [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], physics-guided NNs [18], [19], physics-encoded NNs [20], physics-guided neural-network architectures [21], [22], [23], [24], [25], [26], and physics-inspired neural operators [27], [28] (see [12], [29] for an excellent review). The physics-informed and physics-encoded NNs use partial differential equations (PDEs) for formulating loss functions and/or architectural components, which have deep roots in solving PDEs via NNs [30]. The physics-inspired neural operators [27], [28] have an additional aim of mapping nonlinear functions into alternative domains, where it is easier to train their parameters from observational data and reason about convergence. These seminal frameworks improve the consistency degree with prior analytical knowledge, yet guaranteeing strict compliance still remains challenging.

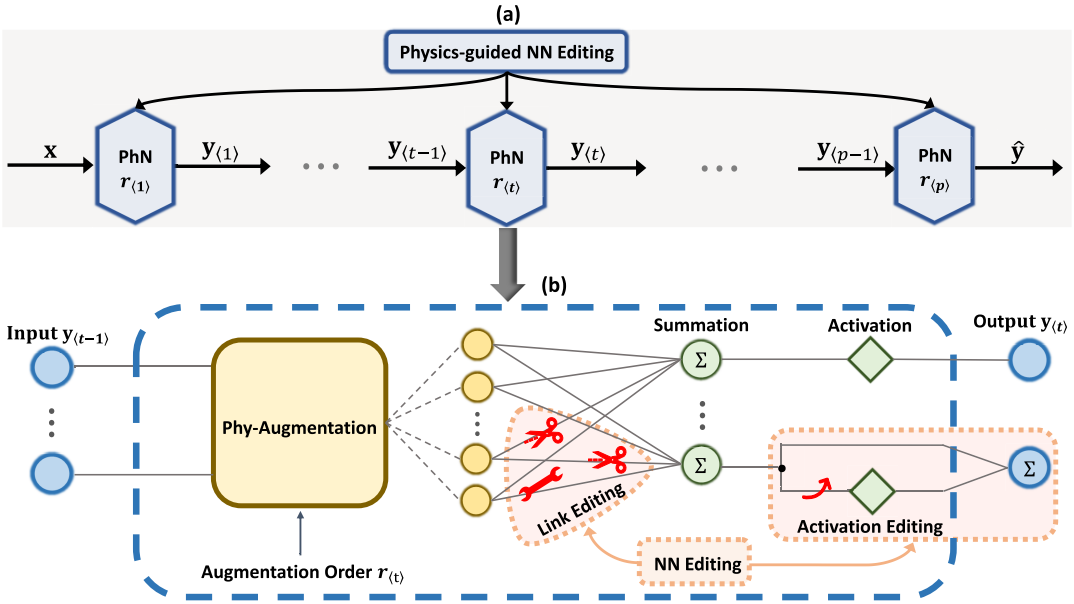

Fig. 1. Architectures of Phy-Taylor and PhN, with NN editing including link and activation editing. (a) Phy-Taylor architecture. (b) PhN architecture.

The physics-encoded NNs are proposed to achieve strict compliance if the prior-known terms in the governing PDEs can be given and formulated as highway filters [20]. However, applying those approaches to physical engineering systems is still challenging, especially when their dynamics are governed by the conjunctive known knowns (e.g., Newton's laws of motion), known unknowns (e.g., Gaussian noise with unknown mean and variance), and unknown unknowns. One critical reason is the known unknowns, and unknown unknowns can result in incomplete PDEs and unavailable prior-known terms of PDEs. Moreover, the fully connected NNs used in those approaches can introduce spurious correlations that deviate from strict compliance with available well-validated physics knowledge. These observations and remaining challenging problems motivate the development of Phy-Taylor. The Phy-Taylor framework leverages the intuition that most physical relations live in low-dimensional manifolds shaped by applicable physical laws. It is just that estimating key physical variables from high-dimensional system observations is often challenging. By expressing known knowledge as relations between yet-to-be-computed latent variables, we force representation learning to converge to a space where these variables represent desired physical quantities shaped by the applicable (expressed) physical laws. We arrive at a desired physics-compliant latent representation by shaping nonlinear terms and relations in the latent space. More specifically, Phy-Taylor offers the following two advantages.

1) *Nonlinear Physics Representation:* Classical NNs can learn arbitrary nonlinear relations by unfolding them into layers of linear weighting functions and switch-like activations. This mechanism is akin to constructing nonlinearities by stitching together piecewise linear behaviors. Instead, by directly exploiting nonlinear terms of the Taylor series expansion, we offer a set of features that express physical nonlinearities much more succinctly, thereby reducing the number of needed parameters and improving the accuracy of representation. Monomials of the Taylor series can capture common nonlinearities present in physics equations, such as kinetic energy, potential energy, rolling resistance, and aerodynamic drag force. The approach constructs input features that represent monomials of the Taylor series and adds a compressor for mitigating the influence of noise on augmented inputs.
2) *Removing Spurious Correlations:* The general topology of NNs allows for models that capture spurious correlations in training samples (overfitting) [31], [32]. In contrast, we develop an NN editing mechanism in the latent space that removes links among certain latent variables when these links contradict their intended physical behaviors, thereby forcing latent representation to converge to variables with the desired semantic interpretation that obeys desired physics relations.

Through experiments with learning dynamics of autonomous vehicles, we show that Phy-Taylor exhibits a considerable reduction in learning parameters, a remarkably accelerated training process, and greatly enhanced model robustness and accuracy (viewed from the perspective of long-horizon trajectory prediction). Experiments with safe velocity regulation in autonomous vehicles further demonstrate that the self-correcting Phy-Taylor successfully addresses the dilemma of prediction horizon and computation time that nonlinear model-predictive control and control barrier function (CBE) are facing in safety-critical control.

Notation: For convenience, Nomenclature summarizes the notations used throughout the article.

II. Problem Formulation

Consider the problem of computing some output vectors, $\mathbf{y}$, from a set of observations, $\mathbf{x}$. The relation between $\mathbf{x}$ and $\mathbf{y}$ is partially determined by physical models of known structure (but possibly unknown parameter values) and partially unknown, thus calling for representation learning of the missing substructures using NN observables. We express the overall input-output relation by the function

$$\mathbf{y} = \underbrace{\mathbf{A}}_{\text{weight matrix}} \cdot \underbrace{\mathfrak{m}(\mathbf{x}, r)}_{\text{node-representation vector}} + \underbrace{\mathbf{f}(\mathbf{x})}_{\text{model mismatch}} \tag{1}$$

where $\mathbf{y}$ and $\mathbf{x}$ are the output and input vectors of the overall system model, respectively, and the parameter $r \in \mathbb{N}$ controls model size. Based on the model formula, we introduce a knowledge set

$$\mathbb{K} \triangleq \{[\mathbf{A}]_{i,j} \big| \frac{\partial [\mathbf{f}(\mathbf{x})]_i}{\partial [\mathfrak{m}(\mathbf{x}, r)]_j} \equiv 0, \quad i \in \{1, \ldots, \text{len}(\mathbf{f}(\mathbf{x}))\} \quad j \in \{1, \ldots, \text{len}(\mathfrak{m}(\mathbf{x}, r))\}\}. \tag{2}$$

The condition in (2) indicates that the knowledge set $\mathbb{K}$ includes: 1) known parameter values but completely unknown model formula; 2) partially known model formula; and 3) known model formula but unknown parameter values. Considering the knowledge set, the problem addressed in this article is formally stated below.

Problem 1: Given a time-series of inputs, $\mathbf{x}$, the corresponding outputs, $\mathbf{y}$, and the knowledge set $\mathbb{K}$ in (2), it is desired to develop an end-to-end NN that directly estimates $\mathbf{y}$ (denoted by $\widehat{\mathbf{y}}$), given $\mathbf{x}$, consistently with all elements of knowledge set $\mathbb{K}$. In other words, the model must satisfy the property that if each $[\mathbf{A}]_{i,j} \in \mathbb{K}$, the end-to-end model must ensure that $((\partial[\widehat{\mathbf{y}}]_i)/(\partial[\mathfrak{m}(\mathbf{x}, r)]_j)) \equiv [\mathbf{A}]_{i,j}$ for any $\mathfrak{m}(\mathbf{x}, r)$.

The above definition allows the system described by (1) to have an end-to-end model that intertwines well-known substructure properties with high-order unmodeled correlations of unknown nonlinear structure. In this, our problem differs from past seminal frameworks of physics-enhanced NNs [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [21], [22], [23], [24], [25], [26], [28], [33], [34], [35], [36], [37], [38], [39], that use the compact PDEs for formulating the PDEs-regulated loss function and/or DNN architectures to count the degree mean of consistency with PDEs. We next use a relatively simple example to explain why the compact governing equations are not available and how to obtain the partially available knowledge.

Example 1 [40]: We let the output of Phy-Taylor approximate the safety metric $V(\mathbf{x}(k), \mathbf{u}(k))$ of ground truth

$$V(\mathbf{x}(k), \mathbf{u}(k)) = \sum_{t=k}^{k+\tau-1} \gamma^{t-k} \cdot \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix}^\top \cdot P \cdot \begin{bmatrix} \mathbf{x}(t) \\ \mathbf{u}(t) \end{bmatrix} \tag{3}$$

where the $\mathbf{x}(k)$ and $\mathbf{u}(k)$ denote the system state and control command at time k, respectively, the P is a positive-definite matrix, the $\gamma \in [0, 1]$ is the discount factor, controlling the relative importance of immediate and future safety metrics. Without loss of generality, we can express state-dependent control policy and real system model as

$$\begin{aligned} \mathbf{u}(k) &= \pi(\mathbf{x}(k)) \\ \mathbf{x}(k+1) &= \widehat{g}(\mathbf{x}(k), \mathbf{u}(k)) = \widehat{g}(\mathbf{x}(k), \quad \pi(\mathbf{x}(k))) \triangleq g(\mathbf{x}(k)) \end{aligned}$$

from which we have

$$\begin{aligned} \mathbf{x}(k+m) &= \underbrace{g \circ g \circ \cdots \circ g}_{m \text{ times operations}} \circ \mathbf{x}(k) = g^m \circ \mathbf{x}(k) \\ \mathbf{u}(k+m) &= \underbrace{\pi \circ g \circ \cdots \circ g}_{m+1 \text{ times operations}} \circ \mathbf{u}(k) = \pi \circ g^m \circ \mathbf{u}(k) \end{aligned}$$

considering which, the safety metric (3) can be rewritten as

$$V(\mathbf{x}(k), \mathbf{u}(k)) = \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix}^\top \cdot P \cdot \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{u}(k) \end{bmatrix} + \sum_{t=k+1}^{k+\tau-1} \gamma^{t-k} \cdot \begin{bmatrix} g^{t-k} \circ \mathbf{x}(k) \\ \pi \circ g^{t-k} \circ \mathbf{x}(k) \end{bmatrix}^\top \cdot P \cdot \begin{bmatrix} g^{t-k} \circ \mathbf{x}(k) \\ \pi \circ g^{t-k} \circ \mathbf{x}(k) \end{bmatrix}$$

observing which we can discover as follows.

1) Due to unknown mappings $g(\cdot)$ and $\pi(\cdot)$, we do not have a compact or precise governing equation of the safety metric at hand.
2) According to Taylor series in (4), one available knowledge about the safety metric formula is the ground-truth $V(\mathbf{x}(k), \mathbf{u}(k))$ does not include any odd-order monomials of $\mathbf{x}(k)$ and $\mathbf{u}(k)$, such as $[\mathbf{u}(k)]_1^3$ and $[\mathbf{u}(k)]_1[\mathbf{x}(k)]_1^2$. Meanwhile, the positive-definite P means the $V(\mathbf{x}(k), \mathbf{u}(k)) \geq 0$. Therefore, the NN editing shall remove the connections with odd-order monomials and maintain some critical connections with even-order monomials to guarantee the nonnegative approximation (i.e., Phy-Taylor's terminal output).

The proposed solution to Problem 1 is the Phy-Taylor framework, presented in Section III.

III. Phy-Taylor Framework

The proposed Phy-Taylor for addressing Problem 1 is depicted in Fig. 1, which is built on the conjunctive deep PhN and physics-guided NN editing. In other words, implementing NN editing according to Taylor's theorem for embedding available physical knowledge into deep PhN (DPhN) yields the Phy-Taylor. We note from Fig. 1 the PhN is a NN layer with a critical component: a physics-inspired augmentation (called Phy-Augmentation) for generating monomials of Taylor series in (1) for capturing nonlinear physical quantities. The physics-guided NN editing—including link editing and activation editing—further modifies network links and activation functions consistently with physics knowledge. Specifically, link editing removes and maintains nodal links according to the consistency with physics knowledge. Meanwhile, the activation editing performs the physics-knowledge-preserving computing in the output channel of each PhN. Collaboratively through link and activation editing, the input–output of Phy-Taylor strictly complies with the available physics knowledge, which is a desired solution to Problem 1. Next, we detail them.

A. Physics-Compatible NN

In order to capture nonlinear features of physical functions, we introduce a new type of network layer that is augmented with terms derived from Taylor series expansion. The Taylor's Theorem offers a series expansion of arbitrary nonlinear functions, as shown below.

Taylor's Theorem (Chapter 2.4 [41]): Let $\mathbf{g}: \mathbb{R}^n \to \mathbb{R}$ be a r-times continuously differentiable function at the point $\mathbf{o} \in \mathbb{R}^n$. Then there exists $\mathbf{h}_\alpha: \mathbb{R}^n \to \mathbb{R}$, where $|\alpha| = r$, such that

$$\mathbf{g}(\mathbf{x}) = \sum_{|\alpha| \leq r} \frac{\partial^\alpha \mathbf{g}(\mathbf{o})}{\alpha!} (\mathbf{x} - \mathbf{o})^\alpha + \sum_{|\alpha| = r} \mathbf{h}_\alpha(\mathbf{x}) (\mathbf{x} - \mathbf{o})^\alpha \quad \text{and} \quad \lim_{\mathbf{x} \to \mathbf{o}} \mathbf{h}_\alpha(\mathbf{x}) = \mathbf{0} \tag{4}$$

where $\alpha = [\alpha_1; \alpha_2; \ldots; \alpha_n]$, $|\alpha| = \sum_{i=1}^n \alpha_i$, $\alpha! = \prod_{i=1}^n \alpha_i!$, $\mathbf{x}^\alpha = \prod_{i=1}^n \mathbf{x}_i^{\alpha_i}$, and $\partial^\alpha \mathbf{g} = ((\partial^{|\alpha|} \mathbf{g})/(\partial \mathbf{x}_1^{\alpha_1} \cdot \ldots \cdot \partial \mathbf{x}_n^{\alpha_n}))$.

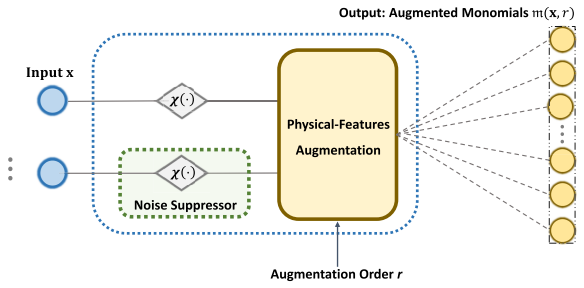

Fig. 2. Phy-Augmentation architecture.

Taylor's theorem has several desirable properties as follows.

1) *Nonlinear Physics Term Representation:* The high-order monomials (i.e., the ones included in $(\mathbf{x}-\mathbf{o})^\alpha$ with $|\alpha| \geq 2$) of the Taylor series (i.e., $\sum_{|\alpha|\leq r}((D^\alpha \mathbf{g}(\mathbf{o}))/\alpha!)(\mathbf{x}-\mathbf{o})^\alpha)$ capture core nonlinearities of physical quantities such as kinetic energy ($\triangleq(1/2)mv^2$), potential energy ($\triangleq(1/2)kx^2$), electrical power ($\triangleq V \cdot I$) and aerodynamic drag force ($\triangleq(1/2)\rho v^2 C_D A$), that drive the state dynamics.
2) *Controllable Model Accuracy:* Given $\mathbf{h}_\alpha(\mathbf{x})$ is finite and $\|\mathbf{x}-\mathbf{o}\| < 1$, the error $\sum_{|\alpha|=r}\mathbf{h}_\alpha(\mathbf{x})(\mathbf{x}-\mathbf{o})^\alpha$ for approximating the ground truth $\mathbf{g}(\mathbf{x})$ will drop significantly as the order $r = |\alpha|$ increases and $\lim_{|\alpha|=r\to\infty}\mathbf{h}_\alpha(\mathbf{x})(\mathbf{x}-\mathbf{o})^\alpha = \mathbf{0}$. This allows for controllable model accuracy via controlling the order r.
3) *Knowledge Embedding:* The Taylor series can directly project the known model substructure parameters of the ground-truth model (1) into NN parameters, including the weight matrix $(((D^\alpha \mathbf{g}(\mathbf{o}))/\alpha!)$ with $|\alpha| > 0)$ and bias $(((D^\alpha \mathbf{g}(\mathbf{o}))/\alpha!)$ with $|\alpha| = 0)$, thus paving the way to embed the available physics knowledge in the form of an appropriately weighted NN layer.

We note that Taylor's theorem relies on the assumption that the ground truth $\mathbf{g}(\mathbf{x})$ is a r-times continuously differentiable function at the point $\mathbf{o}$. If the assumption does not hold, the Taylor series will approximate the proximity of ground truth that is r-times continuously differentiable. For continuous functions, this is often a sufficient approximation.

Next, we describe how PhNs embed the Taylor series expansion into NN layers. The resulting architecture is shown in Fig. 1. Compared with a classical NN layer, we introduce the Phy-Augmentation, whose architecture is shown in Fig. 2. The Phy-Augmentation has two components: 1) physical-features augmentation that generates monomials of a Taylor series expansion and 2) a suppressor for mitigating the influence of noise on such augmented inputs. Next, we detail them.

1) Monomials of Taylor Series: The function of physical-features augmentation in Fig. 2 is to generate the vector of physical features (i.e., node representations) in the form of Taylor series' monomials, which is formally described by Algorithm 1. The Lines 6–13 of Algorithm 1 guarantee that the generated node-representation vector embraces all the nonmissing and nonredundant monomials of the Taylor series. Line 16 shows that Algorithm 1 finally stacks vector with one. This operation means a PhN node will be assigned to be one, and the bias (corresponding to $((D^\alpha \mathbf{g}(\mathbf{o}))/\alpha!)$ with $|\alpha| = 0$ in Taylor series) will be thus treated as link weights in PhN layers. The Phy-Augmentation empowers PhN to capture core nonlinearities of physical quantities (e.g., kinetic energy, potential energy, electrical power, and aerodynamic drag force) that drive the state dynamics and approximate physics knowledge in the form of the Taylor series.

Algorithm 1 Phy-Augmentation Procedure

Input: augmentation order r, input $\mathbf{x}$, point $\mathbf{o}$, suppressor mapping $\chi(\cdot)$.

1 Suppress input:
$$[\mathbf{x}]_i \leftarrow \begin{cases}[\mathbf{x}]_i, & \text{sus} = \text{ina} \\ \chi([\mathbf{x}]_i), & \text{otherwise}\end{cases}, \ i \in \{1, \ldots, \text{len}(\mathbf{x})\};$$
2 Generate index vector of input entries:
$\mathbf{i} \leftarrow [1;\ 2;\ \ldots;\ \text{len}(\mathbf{x})]$;
3 Generate augmentations: $\mathfrak{m}(\mathbf{x}, r) \leftarrow \mathbf{x}$;
4 **for** $\_ = 2$ *to* r **do**
5 **for** $i = 1$ *to* $\text{len}(\mathbf{x})$ **do**
6 Compute temporaries: $\mathbf{t}_a \leftarrow [\mathbf{x}]_i \cdot [\mathbf{x}]_{[[\mathbf{i}]_i\ :\ \text{len}(\mathbf{x})]}$;
7 **if** $i == 1$ **then**
8 Generate temporaries: $\widetilde{\mathbf{t}}_b \leftarrow \widetilde{\mathbf{t}}_a$;
9 **else**
10 Generate temporaries: $\widetilde{\mathbf{t}}_b \leftarrow [\widetilde{\mathbf{t}}_b;\ \widetilde{\mathbf{t}}_a]$;
11 **end**
12 Update index entry: $[\mathbf{i}]_i \leftarrow \text{len}(\mathbf{x})$;
13 Update augmentations:
$\mathfrak{m}(\mathbf{x}, r) \leftarrow [\mathfrak{m}(\mathbf{x}, r);\ \mathbf{t}_b]$;
14 **end**
15 **end**
16 Output vector of augmented monomials:
$\mathfrak{m}(\mathbf{x}, r) \leftarrow [1;\ \mathfrak{m}(\mathbf{x}, r)]$.

We note that Line 1 of Algorithm 1 means the noise suppressor is not applied to all the input elements. The critical reason is the different mapping induced by the suppressor on inputs can destroy the compliance with physical knowledge when the available model-substructure knowledge does not include the mapping of the suppressor.

2) Noise Suppressor: The suppressor in Fig. 2 is to mitigate the influence of noise on the augmented high-order monomials. Before proceeding with the working mechanism, we present noise and true data metrics.

Definition 1: Consider the noisy data and define the data-to-noise ratio (DNR)

$$[\bar{\mathbf{x}}]_i = \underbrace{[\mathbf{h}]_i}_{\text{true data}} + \underbrace{[\mathbf{w}]_i}_{\text{noise}} \in \mathbb{R}, \quad \text{DNR}_i \triangleq \frac{[\mathbf{h}]_i}{[\mathbf{w}]_i}. \tag{5}$$

The auxiliary Theorem 4 presented in Appendix A implies that the high-order monomials can shrink their DNRs due to nonlinear mapping. This means the PhN can be vulnerable to noisy inputs, owning to Phy-Augmentation for generating high-order monomials. Hence, mitigating the influence of noise is vital for enhancing the robustness of PhNs, and consequently the Phy-Taylor. As shown in Fig. 2, we incorporate a suppressor into PhN to process the raw input data, such that the high-order monomial from Phy-Augmentation can enlarge their DNRs. Building on Definition 1, the proposed noise suppressor mapping is

$$\chi([\bar{\mathbf{x}}]_i) = \begin{cases}0, & [\mathbf{h}]_i + [\mathbf{w}]_i < 0 \\ [\mathbf{h}]_i + [\mathbf{w}]_i, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \ \&\ [\mathbf{w}]_i < 0 \\ ([\mathbf{h}]_i + [\mathbf{w}]_i)\cdot\kappa_i + \rho_i, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \ \&\ [\mathbf{w}]_i > 0\end{cases} \tag{6}$$

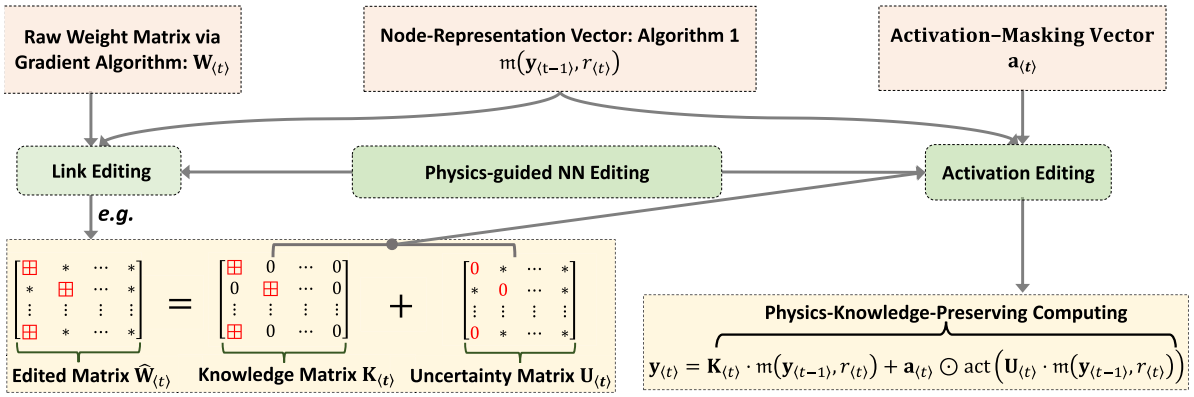

Fig. 3. Flowchart of NN editing in single PhN layer ($\boxplus$ denotes an entry included in knowledge set $\mathbb{K}$). The raw weight matrix $\mathbf{W}_{\langle t\rangle}$ is first generated via gradient descent algorithm. Given this layer's node-representation vector (generated via Algorithm 1) and system matrix $\mathbf{A}$ and knowledge set $\mathbb{K}$, then perform link editing operation, which yields knowledge matrix $\mathbf{K}_{\langle t\rangle}$ and uncertainty matrix $\mathbf{U}_{\langle t\rangle}$ (a direct operation on $\mathbf{W}_{\langle t\rangle}$). The summation of $\mathbf{K}_{\langle t\rangle}$ and $\mathbf{U}_{\langle t\rangle}$ equates the edited weight matrix denoted by $\widehat{\mathbf{W}}_{\langle t\rangle}$. Finally, the resulting $\mathbf{K}_{\langle t\rangle}$ and $\mathbf{U}_{\langle t\rangle}$ are fed to activation editing for performing the knowledge-preserving NN output computation.

where the parameters ρ_i and κ_i satisfy

$$|\rho_i| \geq |[\mathbf{h}]_i + [\mathbf{w}]_i| \cdot |\kappa_i|. \tag{7}$$

We next present the suppressor properties in the following theorem, whose proof appears in Appendix B.

Theorem 1: Consider the noisy data $[\bar{\mathbf{x}}]_i$ and the suppressor described in (5) and (6), respectively. Under the condition (7), the suppressor output, denoted by $[\widehat{\mathbf{x}}]_i = \chi([\bar{\mathbf{x}}]_i)$, has the properties

DNR magnitude of monomial $[\widehat{\mathbf{x}}]_i^p[\widehat{\mathbf{x}}]_j^q$ is strictly increasing with respect to DNR magnitudes of $[\widehat{\mathbf{x}}]_i$ and $[\widehat{\mathbf{x}}]_j$. (8)

The true data and the noise of suppressor output $[\widehat{\mathbf{x}}]_i$ are

$$[\widetilde{\mathbf{h}}]_i = \begin{cases} [\mathbf{h}]_i \cdot \kappa_i + \rho_i, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \text{ and } [\mathbf{w}]_i > 0 \\ [\mathbf{h}]_i, & \text{otherwise} \end{cases} \tag{9}$$

$$[\widetilde{\mathbf{w}}]_i = \begin{cases} -[\mathbf{h}]_i, & [\mathbf{h}]_i + [\mathbf{w}]_i < 0 \\ [\mathbf{w}]_i, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \text{ and } [\mathbf{w}]_i < 0 \\ [\mathbf{w}]_i \cdot \kappa_i, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \text{ and } [\mathbf{w}]_i > 0 \end{cases} \tag{10}$$

such that $[\widehat{\mathbf{x}}]_i = [\widetilde{\mathbf{h}}]_i + [\widetilde{\mathbf{w}}]_i$, $i \in \{1, 2, \ldots, \mathrm{len}(\widehat{\mathbf{x}})\}$.

The result (10) implies the parameters κ and ρ control the DNRs of suppressed data, consequently, the high-order monomials. Furthermore, the result (8) suggests that through designing parameters κ_i, ρ_i, κ_j and ρ_j for increasing the DNR magnitudes of data $[\widehat{\mathbf{x}}]_i$ and $[\widehat{\mathbf{x}}]_j$, the DNR of high-order monomial $[\widehat{\mathbf{x}}]_i^p[\widehat{\mathbf{x}}]_j^q$ can be enlarged consequently, such that the influence of noise is mitigated.

B. Physics-Guided NN Editing

Building on the DPhN, this section presents the NN editing for embedding and preserving the available physics knowledge through link editing and activation editing. Specifically, link editing centers around removing and maintaining the links according to the consistency of physics knowledge. Meanwhile, the activation editing performs the physics-knowledge-preserving computing in the output channels of each PhN layer. Thanks to the concurrent link and activation editing, the input-output of Phy-Taylor can strictly comply with the available physics knowledge. The procedure of physics-guided NN editing is described in Algorithm 2.

Fig. 3 summarizes the flowchart of NN editing in a PhN as follows.

1) Given the node-representation vector from Algorithm 1, the raw weight matrix obtained via gradient descent algorithm is edited via link editing to embed assigned physics knowledge, resulting in an edited matrix denoted by $\widehat{\mathbf{W}}_{\langle t\rangle}$. The edited matrix $\widehat{\mathbf{W}}_{\langle t\rangle}$ can be separated into knowledge matrix $\mathbf{K}_{\langle t\rangle}$ and uncertainty matrix $\mathbf{U}_{\langle t\rangle}$, i.e., $\widehat{\mathbf{W}}_{\langle t\rangle} = \mathbf{K}_{\langle t\rangle} + \mathbf{U}_{\langle t\rangle}$. Specifically, the $\mathbf{K}_{\langle t\rangle}$, generated in Lines 5 and 10, includes all the elements of the knowledge set $\mathbb{K}$ of system matrix of the ground-truth model. While the $\mathbf{M}_{\langle t\rangle}$, generated in Lines 6 and 11, is used to generate uncertainty matrix $\mathbf{U}_{\langle t\rangle}$ (see Line 15) include all the connection weights excluded by the knowledge set $\mathbb{K}$, through freezing the entries of weight matrix $\mathbf{W}_{\langle t\rangle}$ included in the knowledge set $\mathbb{K}$ to zeros.
2) The $\mathbf{K}_{\langle t\rangle}$, $\mathbf{M}_{\langle t\rangle}$ and activation-masking vector $\mathbf{a}_{\langle t\rangle}$ (generated in Lines 7 and 12) are used by activation editing for physics-knowledge-preserving computing of output in each PhN layer. The function of $\mathbf{a}_{\langle t\rangle}$ is to avoid extra (activation) mapping that prior physical knowledge does not include.

If the entries of knowledge matrix $\mathbf{K}_{\langle t\rangle}$ in the same row are all included in the knowledge set, the associated activation should be inactivated. Otherwise, the Phy-Taylor cannot strictly preserve the available physics knowledge due to the extra nonlinear mappings induced by the activation functions. This thus motivates the physics-knowledge-preserving computing, i.e., the Line 16 of Algorithm 2.

Lines 2–6 of Algorithm 2 means that $\mathbf{A} = \mathbf{K}_{\langle 1\rangle} + \mathbf{M}_{\langle 1\rangle} \odot \mathbf{A}$, leveraging which and the setting $r_{\langle 1\rangle} = r$, the ground-truth model (1) is rewritten as

$$\begin{aligned} \mathbf{y} &= (\mathbf{K}_{\langle 1\rangle} + \mathbf{M}_{\langle 1\rangle} \odot \mathbf{A}) \cdot \mathfrak{m}(\mathbf{x}, r) + \mathbf{f}(\mathbf{x}) \\ &= \mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}) + (\mathbf{M}_{\langle 1\rangle} \odot \mathbf{A}) \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}) + \mathbf{f}(\mathbf{x}). \end{aligned} \tag{11}$$

We obtain from the Line 16 of Algorithm 2 that the output of the first PhN layer is

$$\mathbf{y}_{\langle 1\rangle} = \mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}) + \mathbf{a}_{\langle 1\rangle} \odot \mathrm{act}\big(\mathbf{U}_{\langle 1\rangle} \cdot \mathfrak{m}\big(\mathbf{x}, r_{\langle 1\rangle}\big)\big). \tag{12}$$

Recalling the $\mathbf{K}_{\langle 1\rangle}$ includes all the entries of system matrix $\mathbf{A}$ included in the knowledge set $\mathbb{K}$ while the $\mathbf{U}_{\langle 1\rangle}$ includes remainders, we conclude from (11) and (12) that the available

Algorithm 2 Physics-Guided NN Editing

Input: Knowledge set $\mathbb{K}$ (2), terminal output dimension $\text{len}(\mathbf{y})$, number p of PhNs, activation functions $\text{act}(\cdot)$, $\mathbf{y}_{\langle 0\rangle}=\mathbf{x}$ and $r_{\langle 1\rangle}=r$.

1 **for** $t=1$ *to* p **do**
2 **if** $t==1$ **then**
3 Deactivate noise suppressor;
4 Generate node-representation vector $\mathfrak{m}(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle})$ via Algorithm 1;
5 Generate knowledge matrix $\mathbf{K}_{\langle t\rangle}$: $[\mathbf{K}_{\langle t\rangle}]_{i,j} \leftarrow \begin{cases} [\mathbf{A}_{\langle t\rangle}]_{i,j}, & \text{if } [\mathbf{A}_{\langle t\rangle}]_{i,j} \in \mathbb{K} \\ 0, & \text{otherwise} \end{cases}$;
6 Generate weight-masking matrix $\mathbf{M}_{\langle t\rangle}$: $[\mathbf{M}_{\langle t\rangle}]_{i,j} \leftarrow \begin{cases} 0, & \text{if } [\mathbf{A}_{\langle t\rangle}]_{i,j} \in \mathbb{K} \\ 1, & \text{otherwise} \end{cases}$;
7 Generate activation-masking vector $\mathbf{a}_{\langle t\rangle}$: $[\mathbf{a}_{\langle t\rangle}]_i \leftarrow \begin{cases} 0, & \text{if } [\mathbf{A}_{\langle t\rangle}]_{i,j} \in \mathbb{K},\ \forall j \in \{1,\dots,\text{len}(\mathfrak{m}(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle}))\} \\ 1, & \text{otherwise} \end{cases}$;
8 **else**
9 Generate node-representation vector $\mathfrak{m}(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle})$ via Algorithm 1;
10 Generate knowledge matrix $\mathbf{K}_{\langle t\rangle}$:

$$\mathbf{K}_{\langle t\rangle} \leftarrow \left[\begin{array}{c|c|c} \mathbf{0}_{\text{len}(\mathbf{y})} & \mathbf{I}_{\text{len}(\mathbf{y})} & \mathbf{O}_{\text{len}(\mathbf{y})\times(\text{len}(\mathfrak{m}(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle}))-\text{len}(\mathbf{y})-1)} \\ \hline \mathbf{O}_{(\text{len}(\mathbf{y}_{\langle t\rangle})-\text{len}(\mathbf{y}))\times\text{len}(\mathfrak{m}(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle}))} & & \end{array}\right];$$

11 Generate weight-masking matrix
$\mathbf{M}_{\langle t\rangle}$: $[\mathbf{M}_{\langle t\rangle}]_{i,j} \leftarrow \begin{cases} 0, & \frac{\partial[\mathfrak{m}(\mathbf{y}_{\langle t\rangle}, r_{\langle t\rangle})]_j}{\partial[\mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle})]_v} \neq 0 \text{ and } [\mathbf{M}_{\langle 1\rangle}]_{i,v}=0, \quad v \in \{1,2,\dots,\text{len}(\mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}))\} \\ 1, & \text{otherwise} \end{cases}$;
12 Generate activation-masking vector $\mathbf{a}_{\langle t\rangle} \leftarrow \left[\mathbf{a}_{\langle 1\rangle};\ \mathbf{1}_{\text{len}(\mathbf{y}_{\langle t\rangle})-\text{len}(\mathbf{y})}\right]$;
13 **end**
14 Generate raw weight matrix: $\mathbf{W}_{\langle t\rangle}$ via gradient descent algorithm;
15 Generate uncertainty matrix $\mathbf{U}_{\langle t\rangle} \leftarrow \mathbf{M}_{\langle t\rangle} \odot \mathbf{W}_{\langle t\rangle}$;
16 Compute output: $\mathbf{y}_{\langle t\rangle} \leftarrow \mathbf{K}_{\langle t\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle}) + \mathbf{a}_{\langle t\rangle} \odot \text{act}\big(\mathbf{U}_{\langle t\rangle} \cdot \mathfrak{m}\big(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle}\big)\big)$;
17 **end**

Output: terminal output: $\widehat{\mathbf{y}} \leftarrow \mathbf{y}_{\langle p\rangle}$.

physics knowledge about the ground-truth model (1) has been embedded to the first PhN layer. The embedded knowledge shall be passed down to the remaining cascade PhNs and preserved therein, such that the end-to-end Phy-Taylor model can strictly comply with the physics knowledge. This knowledge passing is achieved by the block matrix $\mathbf{K}_{\langle p\rangle}$ generated in Line 10, due to which, the output of tth layer satisfies

$$[\mathbf{y}_{\langle t\rangle}]_{1:\text{len}(\mathbf{y})} = \underbrace{[\mathbf{a}_{\langle t\rangle} \odot \text{act}\big(\mathbf{U}_{\langle t\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle t-1\rangle}, r_{\langle t\rangle})\big)]_{1:\text{len}(\mathbf{y})}}_{\text{knowledge preserving}} + \underbrace{\mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle})}_{\text{knowledge passing}} \quad \forall t \in \{2,3,\dots,p\} \tag{13}$$

where the uncertain matrix computation, i.e., $\mathbf{U}_{\langle t\rangle} = \mathbf{M}_{\langle t\rangle} \odot \mathbf{W}_{\langle t\rangle}$ means the masking matrix $\mathbf{M}_{\langle t\rangle}$ generated in Line 11 can remove the spurious correlations in the cascade PhNs.

C. Solution to Problem 1: Phy-Taylor

As described in Fig. 1, implementing the physics-guided NN editing in the DPhN yields the Phy-Taylor. The Phy-Taylor embeds the available physics knowledge into each PhN layer. Hence, its input-output strictly complies with the physics knowledge, which is formally stated in the following theorem, whose proof appears in Appendix C.

Theorem 2: Consider the Phy-Taylor described by Fig. 1 and the knowledge set $\mathbb{K}$ defined in (2). The input–output (i.e., $\mathbf{x}/\widehat{\mathbf{y}}$) of Phy-Taylor strictly complies with the available knowledge pertaining to the physics model (1) of ground truth, i.e., if the $[\mathbf{A}]_{i,j} \in \mathbb{K}$, then $((\partial[\widehat{\mathbf{y}}]_i)/(\partial[\mathfrak{m}(\mathbf{x}, r)]_j)) \equiv ((\partial[\mathbf{y}]_i)/(\partial[\mathfrak{m}(\mathbf{x}, r)]_j)) \equiv [\mathbf{A}]_{i,j}$ always holds for any $\mathfrak{m}(\mathbf{x}, r)$.

IV. Phy-Taylor Properties

Moving forward, this section focuses on the property analysis of Phy-Taylor.

A. Parameter Quantity Reduction

Fig. 2 shows that the Phy-Augmentation, i.e., the Algorithm 1, expands the input without involving weight-matrix multiplication. This trait can be leveraged to significantly reduce the number of learning parameters, including weights and biases. For the demonstration, we consider the two network models in Fig. 4(a) and (b), where Fig. 4(a) describes a fully connected two-layer network, while Fig. 4(b) describes a single PhN. Observing them, we obtain that given the same dimensions of input and terminal output, the number of learning parameters of Fig. 4(a) is $(m+1)n+(n+1)p$ (including $(m+p)n$ weights and $n+p$ bias), while the number of learning parameters of Fig. 4(b) is $(n+1)p$ (including

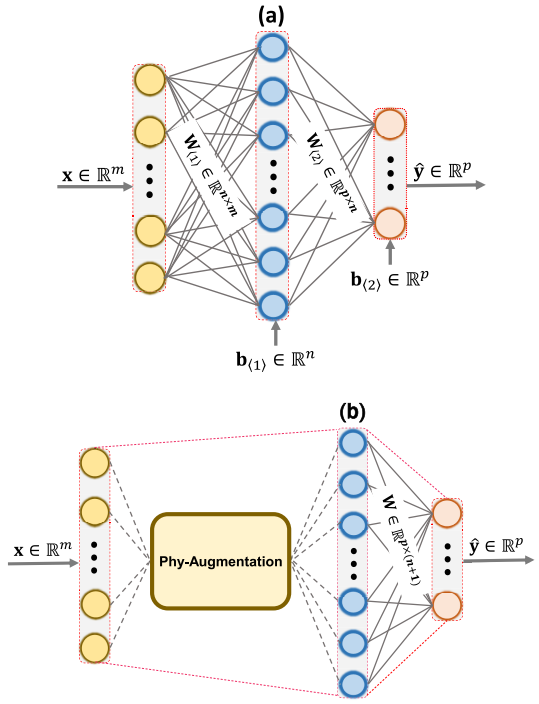

Fig. 4. (a) Two fully connected NN layers. (b) Single PhN layer.

$n \times p$ weights and p bias). The number difference of learning parameters is thus

$$(m+1)n + (n+1)p - (n+1)p = (m+1)n. \tag{14}$$

We note the number of reduced parameters (14) is the lower bound of PhN in the Phy-Taylor framework since it is obtained without considering physics-guided NN editing for removing and freezing links and bias according to the available physics knowledge.

B. Single PhN Versus Cascade PhN

We next investigate if the space complexity (i.e., the number of augmented monomials) of Phy-Augmentation of a single PhN with a large augmentation order can be reduced via cascade PhN with relatively small orders. To simplify the presentation, a single PhN and cascade PhN are, respectively, represented in the following equations:

$$\widehat{\mathbf{y}} = \mathrm{PhN}(\mathbf{x} \in \mathbb{R}^n \big| r) \in \mathbb{R}^m \tag{15}$$

$$\begin{aligned}\mathbf{x} \in \mathbb{R}^n &\longmapsto \mathbf{y}_{\langle 1\rangle} = \mathrm{PhN}(\mathbf{x}|r_{\langle 1\rangle}) \in \mathbb{R}^{n_{\langle 1\rangle}} \\ &\longmapsto \cdots \longmapsto \mathbf{y}_{\langle d-1\rangle} = \mathrm{PhN}(\mathbf{y}_{\langle d-2\rangle}\big|r_{\langle d-1\rangle}) \in \mathbb{R}^{n_{\langle d-1\rangle}} \\ &\longmapsto \widehat{\mathbf{y}} = \mathrm{PhN}(\mathbf{y}_{\langle d-1\rangle}\big|r_{\langle d\rangle}) \in \mathbb{R}^m \end{aligned} \tag{16}$$

where the cascade architecture consists of d PhNs. To guarantee the cascade PhN (16) and the single PhN (15) have the same monomials, their augmentation orders shall satisfy

$$\prod_{v=1}^{d} r_{\langle v\rangle} = r \quad \forall r_{\langle v\rangle},\ r \in \mathbb{N}. \tag{17}$$

The space complexity difference of Phy-Augmentation is formally presented in the following theorem, whose proof is presented in Appendix D.

Theorem 3: Under the condition (17), the space complexity difference between single PhN (15) and cascade PhN (16), due to Phy-Augmentation, is

$$\begin{aligned}&\mathrm{len}(\mathfrak{m}(\mathbf{x}, r)) - \sum_{p=1}^{d} \mathrm{len}(\mathfrak{m}(\mathbf{x}, r_{\langle p\rangle})) \\ &= \sum_{s=r_{\langle 1\rangle}+1}^{r} \frac{(n+s-1)!}{(n-1)!s!} - \sum_{v=1}^{d-1}\sum_{s=1}^{r_{\langle v+1\rangle}} \frac{(n_{\langle v\rangle}+s-1)!}{(n_{\langle v\rangle}-1)!s!} \\ &\quad + 1 - d. \end{aligned} \tag{18}$$

The Theorem 3 implies that the output dimensions and the augmentation orders of intermediate PhNs are critical in significantly reducing space complexity via cascade PhN. However, an intuitive question arises: *Does the cascade PhN reduce the complexity at the cost of model accuracy?* The intuitive answer is *if the reduced weights are associated with the links that contradict physics knowledge, the cascade PhN can further increase model accuracy. Otherwise, it can reduce the space complexity at the cost of model accuracy.*

V. Extension: Self-Correcting Phy-Taylor

Safe control and planning is a fundamental solution for enhancing safety assurance of the physical engineering systems often operating in environments where time and safety are critical, such as airplanes, medical drones, and autonomous vehicles. To comply with safety constraints in the face of potential conflicts from control objectives, the CBF framework has been proposed to compute real-time safety-critical control commands [42], [43]. The CBFs, however, use only current state information without prediction, whose control policy is thus greedy and challenging for proactive safe control. It is well known that model predictive control (MPC) yields a less greedy safe control policy since it takes future state information into account [44], [45]. Motivated by the observations, MPC with the incorporation of CBF, i.e., MPC–CBF, was proposed [46]. Due to the nonlinear dynamics, the MPC–CBF faces a dilemma of prediction horizon and computation time of safe control commands, which induces considerable feedback delays and thus leads to failures in the time- and safety-critical operating environments. To address the dilemma, we propose the self-correcting Phy-Taylor, whose architecture is shown in Fig. 5. Its one mission is learning the safety relationship between real-time decisions and safety metrics with consideration of future information

$$\mathbf{s}(\mathbf{x}(k), \mathbf{u}(k), \tau) = \sum_{t=k}^{k+\tau-1} \tilde{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t)) \tag{19}$$

where $\tilde{\mathbf{f}}(\mathbf{x}(t), \mathbf{u}(t))$ is the predefined vector of safety metrics at time t, and τ denotes the horizon of future information of safety metrics.

Inside the self-correcting Phy-Taylor, the learned safety relationship for approximating (19) will first be subject to the off-line verification of available physics knowledge, based on which the necessary revisions can be needed. According to the off-line verified and revised (if needed) safety relationship, the correcting of real-time decision $\mathbf{u}(k)$ will be triggered if any safety metric $[\mathbf{s}(\mathbf{x}(k), \mathbf{u}(k), \tau)]_i$, $i \in \{1, 2, \ldots, h\}$, exceeds (or leaves) the preset safety bounds (or safety envelopes). However, the current learned formula corresponding to (19)

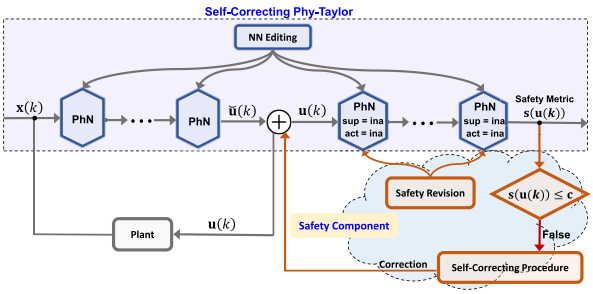

Fig. 5. Self-correcting Phy-Taylor architecture: $\mathbf{u}(k)$ denotes the vector of real-time decisions, $\mathbf{s}(u(k))$ denotes the vector of real-time safety metrics, $\mathbf{c}$ is the vector of safety bounds.

is not ready (if not impossible) for delivering the procedure, owing to the complicated dependence of $[\mathbf{s}(\mathbf{x}(k), \mathbf{u}(k), \tau)]_i$ on both system state $\mathbf{x}(k)$ and control command $\mathbf{u}(k)$. To address this problem, as shown in Fig. 5, we decouple the real-time commands from the real-time system states.

1) Given the real-time system state $\mathbf{x}(k)$ as the origin input, the first Phy-Taylor outputs the real-time control command $\mathbf{u}(k)$, which is motivated by the fact that the state-feedback control is used most commonly in physical engineering systems [47].
2) Given the real-time control command $\mathbf{u}(k)$ (i.e., the output of the first Phy-Taylor) as the input of the second Phy-Taylor, the terminal output is the real-time safety metric $\mathbf{s}(\mathbf{u}(k))$, which is motivated by the fact that the control command $\mathbf{u}(k)$ manipulates system state. In other words, the safety metric $\mathbf{s}(\mathbf{u}(k))$ directly depends on $\mathbf{u}(k)$ while indirectly depends on $\mathbf{x}(k)$.
3) The two Phy-Taylors are trained simultaneously according to the training loss function

$$\mathcal{L} = \alpha \|\mathbf{s}(\mathbf{u}(k)) - \mathbf{s}(\mathbf{x}(k), \mathbf{u}(k), \tau)\| + \beta \left\| \check{\mathbf{u}}(k) - \mathbf{u}(\mathbf{k}) \right\|$$

where the $\mathbf{s}(\mathbf{x}(k), \mathbf{u}(k), \tau)$ given in (19) is ground truth of safety-metric vector, the $\check{\mathbf{u}}(k)$ is ground truth of control command, the α and β are hyperparameters. The two cascade Phy-Taylors thus depend on each other.
4) To render the learned safety relationship $\mathbf{s}(\mathbf{u}(k))$ tractable, activations and compressors of second Phy-Taylor are inactive.

Given the verified and revised (if needed) relationship, the self-correcting procedure will be triggered (if exceeding safety bound $\mathbf{c}$) for correcting control command according to

$$\mathbf{u}(k) \leftarrow \underset{\tilde{\mathbf{u}}(k) \in \mathbb{R}^m}{\arg\min} \ \{ \|\tilde{\mathbf{u}}(k) - \mathbf{u}(k) \| | \mathbf{s}(\tilde{\mathbf{u}}(\mathbf{k})) < \mathbf{c} \}. \quad (20)$$

We note the self-correcting mechanism and the safety revision of the relationship between $\mathbf{s}(\mathbf{u}(k))$ and $\mathbf{u}(k)$ for delivering (20) vary with safety problems and physical systems. An example in this article is the safe control of autonomous vehicles: Algorithm 3 in Section VI.

VI. Experiments

The experiment of this article performs the demonstration of two functions: 1) the learning of the vehicle's conjunctive lateral and longitudinal dynamics via Phy-Taylor and 2) the safe velocity regulation via self-correcting Phy-Taylor. The vehicle operates in the AutoRally platform [48]. The demonstration video is available at https://www.youtube.com/watch?v = dABt_nIBsdQ. Due to the page limit, the experiments demonstrating the effectiveness of noise suppressors in mitigating the influence of noise, and the experiments of embedding different degrees of physical knowledge are presented in our ArXiv reference [49].

A. Vehicle Dynamics Learning

We first identify the available physics knowledge for the physics-guided NN editing. According to Newton's second law for motion along longitudinal and lateral axes [50], we have the following governing equations:

$$\begin{aligned} \bar{m}\ddot{\mathrm{p}} &= F_{\mathrm{p}f} + F_{\mathrm{p}r} - F_{\mathrm{aero}} - R_{\mathrm{p}f} - R_{\mathrm{p}r} \\ \bar{m}(\ddot{\mathrm{y}} + \dot{\psi} v_{\mathrm{p}}) &= F_{\mathrm{y}f} + F_{\mathrm{y}r} \end{aligned}$$

where p is the longitudinal position, y is the lateral position, ψ is the vehicle yaw angle, $\bar{m}$ is the vehicle mass, $v_{\mathrm{p}} \triangleq \dot{\mathrm{p}}$ is the longitudinal velocity, $F_{\mathrm{p}f}$ and $F_{\mathrm{p}r}$ denote the longitudinal tire force at the front and rear tires, respectively, $R_{\mathrm{p}f}$ and $R_{\mathrm{p}r}$ denote the rolling resistance at the front and rear tires, respectively, F_{aero} represents the longitudinal aerodynamic drag force, $F_{\mathrm{y}f}$ and $F_{\mathrm{y}r}$ are the lateral tire forces of the front and rear wheels, respectively. Defining the lateral velocity $v_{\mathrm{y}} \triangleq \dot{\mathrm{y}}$ and yaw velocity $v_{\psi} \triangleq \dot{\psi}$, the following state space model is derived from the above force balance equation [50]:

$$\frac{d}{dt}\begin{bmatrix} \mathrm{p} \\ \mathrm{y} \\ \psi \\ v_{\mathrm{p}} \\ v_{\mathrm{y}} \\ v_{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & * & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & 0 & * & * \end{bmatrix} \underbrace{\begin{bmatrix} \mathrm{p} \\ \mathrm{y} \\ \psi \\ v_{\mathrm{p}} \\ v_{\mathrm{y}} \\ v_{\psi} \end{bmatrix}}_{\triangleq \mathbf{x}} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ * \\ 0 \\ 0 \end{bmatrix} \theta + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ * \\ * \end{bmatrix} \delta$$

where "*" can represent a state-dependent or time-dependent function or a mix of them or just a scalar, but is unknown to us, and θ and δ denote throttle and steering, respectively. Given the practical physics knowledge that *the throttle computation depends on the longitudinal velocity and position only, while the dependencies of steering are unknown*, the state space model above updates with

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ * & 0 & 0 & * & 0 & 0 \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{bmatrix} \mathbf{x}.$$

The sampling technique, with a sampling period denoted by T, converts the continuous-time state–space model above to the following discrete-time one:

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ * & 0 & 0 & * & 0 & 0 \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{bmatrix} \mathbf{x}(k). \quad (21)$$

We first consider two Phy-Taylor models, named "Phy-Taylor$_{\text{large order}}$" and "Phy-Taylor$_{\text{small order}}$," which can embed the available knowledge [i.e., the known parameters included in system matrix of model (21)]. Their architectures are shown in Fig. 6(a) and (b). The Phy-Taylor$_{\text{large order}}$ has one PhN with a large augmentation order while the Phy-Taylor$_{\text{small order}}$

TABLE I
MODEL CONFIGURATIONS

	Layer 1		Layer 2		Layer 3			
Model ID	#weights	#bias	#Weights	#bias	#weights	#bias	#parameter sum	prediction error e
$\text{DPhN}_{\text{large order}}$	3135	15	90	6	–	–	3246	nan
$\text{DPhN}_{\text{small order}}$	270	10	520	8	48	6	862	45.57277
$\text{Phy-Taylor}_{\text{large order}}$	2313	12	30	3	–	–	2358	0.047758
$\text{Phy-Taylor}_{\text{small order}}$	167	7	265	5	18	3	465	0.003605
	Encoder		Decoder		Auxiliary Networks			
Model ID	#weights	#bias	#weights	#bias	#weights	#bias	#parameter sum	prediction error
Deep Koopman	2040	176	2040	176	19920	486	24838	0.232236

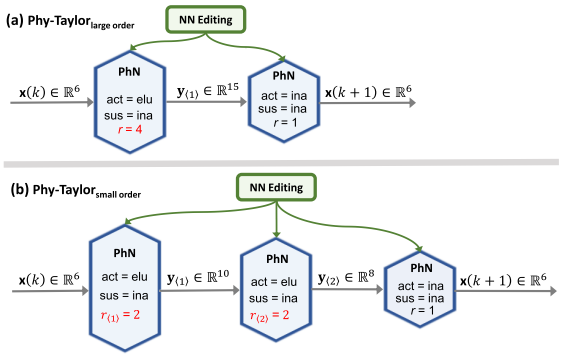

Fig. 6. (a) $\text{Phy-Taylor}_{\text{large order}}$ has a PhN with large order $r = 4$. (b) $\text{Phy-Taylor}_{\text{small order}}$ has cascading PhNs with relatively small orders satisfying $r_{(1)} \cdot r_{(2)} = 2 \cdot 2 = 4 = r$.

has two cascade PhN layers with two relatively small augmentation orders. Meanwhile, the three orders satisfy the condition (17) for having the same monomials of the Taylor series. We also consider the corresponding models without NN editing (i.e., without physics knowledge embedding), which degrades the Phy-Taylor to the DPhN. The two DPhN models are named "$\text{DPhN}_{\text{large order}}$" and "$\text{DPhN}_{\text{small order}}$." The final model we considered is the seminal Deep Koopman [28], following the same model configurations therein. The configurations of five models are summarized in Table I.

The trajectories of training loss are presented in Fig. 7(a)–(c). The (training loss, validation loss) of trained $\text{DPhN}_{\text{large order}}$, $\text{DPhN}_{\text{small order}}$, $\text{Phy-Taylor}_{\text{large order}}$ and $\text{Phy-Taylor}_{\text{small order}}$ are (0.00389, 0.00375), (0.000344, 0.000351), (0.000222, 0.000238) and (0.000915, 0.000916), respectively. To perform the testing, we consider the long-horizon prediction of system trajectories, given the same initial conditions. The prediction error is measured by the mean squared error: $e = (1/\kappa)\sum_{t=k+1}^{k+\kappa}(1/6)\|\underline{\mathbf{x}}(t) - \mathbf{x}(t)\|$ with $\underline{\mathbf{x}}(k) = \mathbf{x}(k)$, where $\underline{\mathbf{x}}(t)$ is the prediction of ground truth $\mathbf{x}(t)$ at time t. The prediction errors over the horizon $\kappa = 300$ and initial time $k = 950$ are summarized in Table I. The ground-truth trajectories and predicted ones from Deep Koopman and the Phy-Taylor models are presented in Fig. 7(d)–(g) 7(i)–(vi). Observing from Table I and Fig. 7, we can conclude:

1) Fig. 7(a) and (b) and (d)–(g): the physics-guided NN editing can significantly accelerate model training, reduce validation and training loss as well as improve model accuracy (viewed from long-horizon prediction of trajectory).
2) Fig. 7(c) and (d)–(g): with physics-guided NN editing, the cascade PhN with small augmentation orders can further significantly reduce training loss, and increase model accuracy. This can be due to the further removed spurious correlations or NN links contradicting with physics law, via the cascade architecture.
3) Fig. 7(d)–(g) and Table I: compared with the fully-connected DPhN models, i.e., $\text{DPhN}_{\text{large order}}$ and $\text{DPhN}_{\text{small order}}$, the seminal Deep Koopman strikingly increases model accuracy, viewed from the perspective of long-horizon prediction of trajectory. Compared to Deep Koopman, the Phy-Taylor models (both $\text{Phy-Taylor}_{\text{large order}}$ and $\text{Phy-Taylor}_{\text{small order}}$) notably reduce the model learning parameters (weights and bias) and further remarkably increase model accuracy simultaneously.

B. Self-Correcting Phy-Taylor

This experiment demonstrates the effectiveness of self-correcting Phy-Taylor in guaranteeing a vehicle's safe driving. The architecture of self-correcting Phy-Taylor is presented in Fig. 8. Its real-time input vector is $\mathbf{x}(k) = [w(k);\ \text{p}(k);\ \text{y}(k);\ \psi(k);\ v_{\text{p}}(k);\ v_{\text{y}}(k);\ v_{\psi}(k)]$, where $w(k)$ is the average of four wheels' velocities. The mediate output $\mathbf{u}(k) = [\theta(k);\ \gamma(k)]$ denotes the vector of control commands, where $\theta(k) \in [-0.156,\ 0.156]$ is the throttle command and $\gamma(k) \in [-0.6,\ 0.6]$ is the steering command. The considered safety-metric vector in (19) is

$$\mathbf{s}(\mathbf{x}(k), \mathbf{u}(k), \tau) = \sum_{t=k+1}^{k+\tau} \left[\left(v_{\text{p}}(t) - \text{v}\right)^2;\ \left(v_{\text{p}}(t) - r \cdot w(k)\right)^2 \right]$$

where v and r denote the reference of longitudinal velocity and the wheel radius, respectively. The safety metrics indicate the objective of the safe control command is to simultaneously steer the vehicle's longitudinal velocity to reference v and constrain the slip (i.e., $(v_{\text{x}}(t) - r \cdot w(k))^2$) to prevent slipping and sliding. The hyperparameters of the training loss function are set to $\alpha = \beta = 1$.

We output the learned safety relationships for off-line verification and necessary revision

$$[\mathbf{s}(\mathbf{u})]_1 = 0.00111 + \begin{bmatrix} \theta \\ \gamma \end{bmatrix}^{\top} \begin{bmatrix} -0.04581 & 0.00100 \\ 0.00100 & 0.00342 \end{bmatrix} \begin{bmatrix} \theta \\ \gamma \end{bmatrix} \tag{22a}$$

$$[\mathbf{s}(\mathbf{u})]_2 = 0.14376 - \begin{bmatrix} \theta \\ \gamma \end{bmatrix}^{\top} \begin{bmatrix} 6.06750 & 0.02701 \\ 0.02701 & 0.00601 \end{bmatrix} \begin{bmatrix} \theta \\ \gamma \end{bmatrix}. \tag{22b}$$

The safety metrics of ground truth are always nonnegative. We thus need to verify that given the ranges of control commands (i.e., $\theta(k) \in [-0.156, 0.156]$, $\gamma(k) \in [-0.6, 0.6]$,

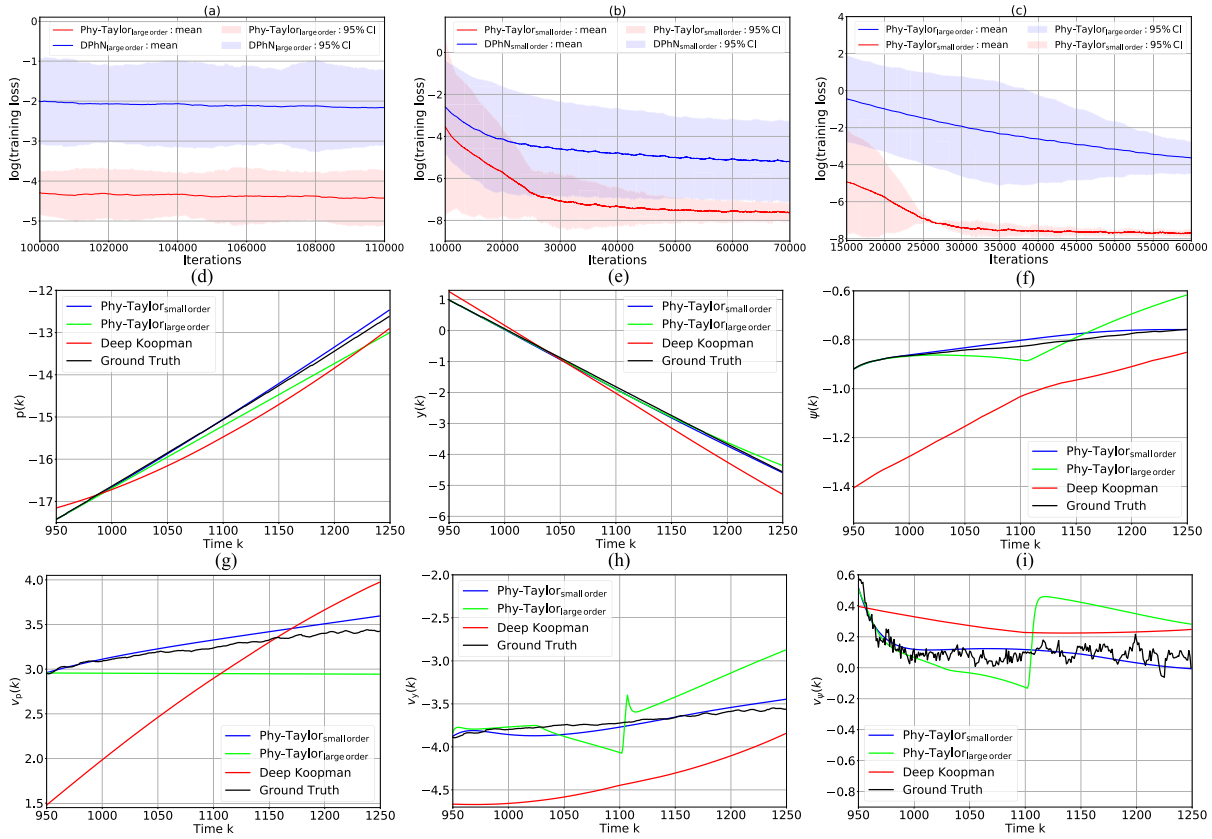

Fig. 7. Training and testing. (a)–(c) Trajectories of averaged training loss (five random seeds) of different models described in Table I. (d)–(i) Ground truth and predicted trajectories via trained models.

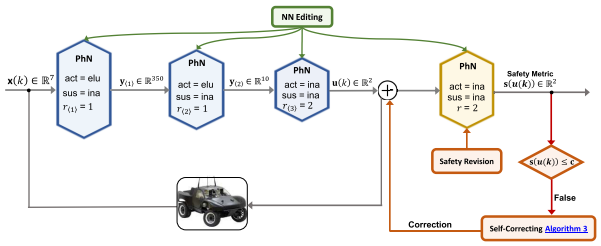

Fig. 8. Self-correcting Phy-Taylor for safe control of autonomous vehicle.

$\forall k \in \mathbb{N}$), if both $[\mathbf{s}(\mathbf{u}(k))]_1$ and $[\mathbf{s}(\mathbf{u}(k))]_2$ in (22) can always be nonnegative. If a violation occurs, we will make revisions to the relationships. We can verify from (22) that the nonnegativity constraint does not always hold, such as $\theta(k) = 0.156$ and $\gamma(k) = 0$. Therefore, the revision of the safety relationship is needed before working on the self-correcting procedure. The regulated safety relationships (revisions are highlighted in red color) are presented below

$$[\mathbf{s}(\mathbf{u})]_1 = \underbrace{0.00021}_{[\mathbf{b}]_1} + \begin{bmatrix}\theta \\ \gamma\end{bmatrix}^\top \underbrace{\begin{bmatrix}0.00181 & 0.00100 \\ 0.00100 & 0.00342\end{bmatrix}}_{\triangleq \mathbf{P}_1} \begin{bmatrix}\theta \\ \gamma\end{bmatrix} \tag{23a}$$

$$[\mathbf{s}(\mathbf{u})]_2 = \underbrace{0.14376}_{[\mathbf{b}]_2} - \begin{bmatrix}\theta \\ \gamma\end{bmatrix}^\top \underbrace{\begin{bmatrix}5.90769 & 0.01201 \\ 0.01201 & 0.00601\end{bmatrix}}_{\triangleq \mathbf{P}_2} \begin{bmatrix}\theta \\ \gamma\end{bmatrix} \tag{23b}$$

which satisfy $[\mathbf{s}(\mathbf{u}(k))]_1 \geq 0$ and $[\mathbf{s}(\mathbf{u}(k))]_2 \geq 0$, for any $\theta(k) \in [-0.156, 0.156]$ and $\gamma(k) \in [-0.6, 0.6]$.

We now are ready to develop the self-correcting procedure. Considering the two matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ defined in (23) are symmetric, we have

$$\mathbf{P}_1 = \underbrace{\begin{bmatrix}-0.934 & -0.3572 \\ -0.3572 & 0.934\end{bmatrix}}_{\triangleq \mathbf{Q}_1} \begin{bmatrix}\underbrace{0.0008}_{\triangleq \lambda_1} & 0 \\ 0 & \underbrace{0.0038}_{\triangleq \lambda_2}\end{bmatrix} \times \underbrace{\begin{bmatrix}-0.934 & -0.3572 \\ -0.3572 & 0.934\end{bmatrix}}_{=\mathbf{Q}_1^\top} \tag{24}$$

$$\mathbf{P}_2 = \mathbf{Q}_1 \cdot \mathbf{Q}_1 \cdot \mathbf{P}_2 \cdot \mathbf{Q}_1 \cdot \mathbf{Q}_1 \tag{25}$$

based on which we further define

$$\widehat{\mathbf{u}}(k) \triangleq \mathbf{Q}_1 \begin{bmatrix}\theta(k) \\ \gamma(k)\end{bmatrix}, \quad \mathbf{S} \triangleq \mathbf{Q_1} \cdot \mathbf{P_2} \cdot \mathbf{Q_1} = \begin{bmatrix}s_{11} & s_{12} \\ s_{12} & s_{22}\end{bmatrix}. \tag{26}$$

We let $[\widehat{\mathbf{c}}]_1$ and $[\widehat{\mathbf{c}}]_2$ denote the two assigned safety metrics. According to the derivations appearing in Appendix E, the control commands included in the safety formulas $[\mathbf{s}(\mathbf{u}(k))]_1 = [\widehat{\mathbf{c}}]_1$ and $[\mathbf{s}(\mathbf{u}(k))]_2 = [\widehat{\mathbf{c}}]_2$ are obtained as

$$\begin{bmatrix}\pm\widehat{\theta}(k) \\ \pm\widehat{\gamma}(k)\end{bmatrix}$$

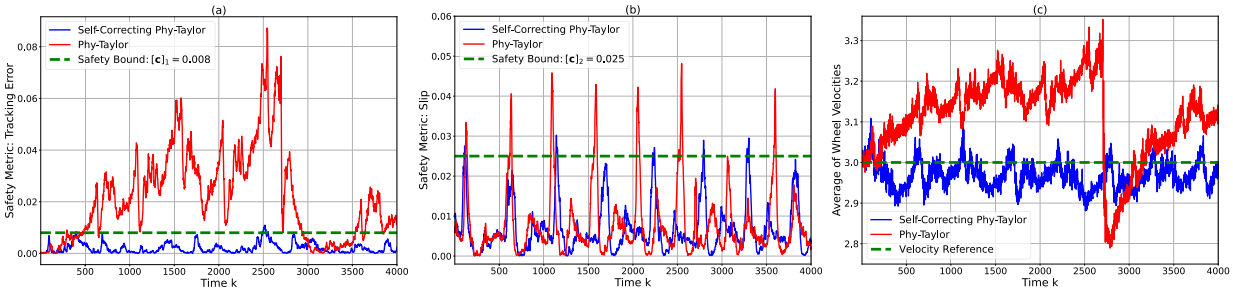

Fig. 9. (a) Safety metric: tracking error (tracking error $(v_p(t) - v)^2$. (b) Safety metric: wheel slip $(v_p(t) - r \cdot w(k))^2$). (c) Average wheel velocities.

$$\triangleq \mathbf{Q}_1 \begin{bmatrix} \pm\sqrt{\dfrac{[\widehat{\mathbf{c}}]_1 - [\mathbf{b}]_1}{\lambda_1} - \dfrac{\lambda_2}{\lambda_1}\dfrac{\sqrt{\varpi_2^2 - 4\varpi_1\varpi_3} - \varpi_2}{2\varpi_1}} \\ \pm\sqrt{\dfrac{\sqrt{\varpi_2^2 - 4\varpi_1\varpi_3} - \varpi_2}{2\varpi_1}} \end{bmatrix} \tag{27}$$

where as in (28)–(30), shown at the bottom of the page.

Solution (27) has paved the way to delivering the self-correcting procedure, i.e., Algorithm 3. The algorithm can be summarized as if the real-time safety metric $[\mathbf{s}(\mathbf{u}(k))]_1$ or. $[\mathbf{s}(\mathbf{u}(k))]_2$ is larger than the corresponding safety bound $[\mathbf{c}]_1$ or $[\mathbf{c}]_2$, the real-time safety metric will be updated with the corresponding safety bound (indicated by Line 4 of Algorithm 3). The corrected control commands are then computed according to (27) (see Lines 8–10). The solutions, however, are not unique. To address the problem, Line 11 of Algorithm 3 picks up the control commands that are most close to the current ones.

Under the control of Phy-Taylor, with and without the self-correcting procedure, the system performances are presented in Fig. 9(a)–(c), which shows the self-correcting approach can significantly enhance safety assurance.

C. Code and Training

For the code, we use the Python API for the TensorFlow framework [51] and the Adam optimizer [52] for training. The Python version is 2.7.12. The TensorFlow version is 1.14.0. Our source code is publicly available at GitHub: https://github.com/ymao578/Phy-Taylor.

We set the batch-size to 200 for Sections VI-A and VI-B. Their learning rates are set to 0.0005 and 0.00005, respectively. In all the experiments, each weight matrix is initialized randomly from a (truncated) normal distribution with zero mean and standard deviation, discarding and re-drawing any samples more than two standard deviations from the mean. We initialize each bias according to the normal distribution with zero mean and standard deviation. All the models are trained for 10^6 steps.

Algorithm 3 Self-Correcting Procedure

Input: Real-time control-command vector $\mathbf{u}(k) = [\theta(k);\ \gamma(k)]$, safety bounds $[\mathbf{c}]_1$ and $[\mathbf{c}]_2$, and learned matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ and bias $[\mathbf{b}]_1$ and $[\mathbf{b}]_2$ defined in (23).

1 Update original safety relationship with off-line verified and revised one: $\mathbf{s}(\mathbf{u}(\mathbf{k})) \leftarrow$ (23);
2 **if** $[\mathbf{s}(\mathbf{u}(k))]_1 > [\mathbf{c}]_1$ *or* $[\mathbf{s}(\mathbf{u}(k))]_2 > [\mathbf{c}]_2$ **then**
3 **if** $[\mathbf{s}(\mathbf{u}(k))]_i \geq [\mathbf{c}]_i,\ i \in \{1, 2\}$ **then**
4 Update safety metric: $[\widehat{\mathbf{c}}]_i \leftarrow [\mathbf{c}]_i, i \in \{1, 2\}$;
5 **else**
6 Maintain safety metric: $[\widehat{\mathbf{c}}]_i \leftarrow [\mathbf{s}(\mathbf{u}(k))]_i, i \in \{1, 2\}$;
7 **end**
8 Compute orthogonal matrix $\mathbf{P}_1$ and eigenvalues λ_1 and λ_1 according to (24);
9 Compute matrix: $\mathbf{S} \leftarrow \mathbf{Q}_1 \cdot \mathbf{P}_2 \cdot \mathbf{Q}_1$;
10 Compute $\widehat{\theta}(k)$ and $\widehat{\gamma}(k)$ according to (27);
11 Correct real-time control commands:

$$\theta(k) \leftarrow \underset{\{\widehat{\theta}(k), -\widehat{\theta}(k)\}}{\arg\min} \left\{|\theta(k) - \widehat{\theta}(k)|, |\theta(k) + \widehat{\theta}(k)|\right\}$$
$$\gamma(k) \leftarrow \underset{\{\widehat{\gamma}(k), -\widehat{\gamma}(k)\}}{\arg\min} \{|\gamma(k) - \widehat{\gamma}(k)|, |\gamma(k) + \widehat{\gamma}(k)|\}.$$

12 **else**
13 Maintain real-time control commands: $\theta(k) \leftarrow \theta(k)$ and $\gamma(k) \leftarrow \gamma(k)$.
14 **end**

VII. Discussion

In this article, we have proposed a physics-knowledge-enhanced DNN framework called Phy-Taylor. The Phy-Taylor framework introduces two contributions: the deep PhNs and a physics-guided NN editing mechanism to ensure strict

$$\varpi_1 \triangleq \left(\frac{\lambda_2}{\lambda_1}\right)^2 s_{11}^2 + s_{22}^2 + \frac{(4s_{12}^2 - 2s_{11}s_{22})\lambda_2}{\lambda_1} \tag{28}$$

$$\varpi_2 \triangleq -\frac{2([\widehat{\mathbf{c}}]_1 - [\mathbf{b}]_1)\lambda_2 s_{11}^2}{\lambda_1^2} - 2([\mathbf{b}]_2 - [\widehat{\mathbf{c}}]_2)s_{22} + \frac{2([\widehat{\mathbf{c}}]_1 - [\mathbf{b}]_1)s_{11}s_{22} - 4([\widehat{\mathbf{c}}]_1 - [\mathbf{b}]_1)s_{12}^2 + 2\lambda_2([\mathbf{b}]_2 - [\widehat{\mathbf{c}}]_2)s_{11}}{\lambda_1} \tag{29}$$

$$\varpi_3 \triangleq -\frac{2([\widehat{\mathbf{c}}]_1 - [\mathbf{b}]_1)([\mathbf{b}]_2 - [\widehat{\mathbf{c}}]_2)s_{11}}{\lambda_1} + ([\mathbf{b}]_2 - [\widehat{\mathbf{c}}]_2)^2 + \left(\frac{[\widehat{\mathbf{c}}]_1 - [\mathbf{b}]_1}{\lambda_1}\right)^2 s_{11}^2. \tag{30}$$

compliance with prior physics knowledge. As an extension, we have also proposed a self-correcting Phy-Taylor framework that introduces a core capability of automatic output correction when a safety violation occurs.

The current Phy-Taylor can suffer from the curse of dimensionality, so it can hardly be applied to high-dimensional data, such as images and text. The tensor decomposition has the potential to address this problem since it can decompose the higher-order derivatives in Taylor expansions parameterized by DNNs into small core tensors and a set of factor matrices.

Appendix A
Auxiliary Theorems

Theorem 4: The DNR magnitude of high-order monomial $[\bar{\mathbf{x}}]_i^p[\bar{\mathbf{x}}]_j^q$, $p, q \in \mathbb{N}$, is strictly increasing with respect to $|\mathrm{DNR}_i|$ and $|\mathrm{DNR}_j|$, if

$$\mathrm{DNR}_i, \mathrm{DNR}_j \in (-\infty, -1] \text{ or } \mathrm{DNR}_i, \mathrm{DNR}_j \in \left[-\frac{1}{2}, 0\right) \text{ or } \mathrm{DNR}_i, \mathrm{DNR}_j \in (0, \infty). \quad (31)$$

Proof: In view of Definition 1, the true data can be equivalently expressed as $[\mathbf{h}]_i = \mathrm{DNR}_i \cdot [\mathbf{w}]_i$, according to which we have $[\bar{\mathbf{x}}]_i = (1 + \mathrm{DNR}_i)[\mathbf{w}]_i$ such that

$$\begin{aligned} [\bar{\mathbf{x}}]_i^p[\bar{\mathbf{x}}]_j^q &= (1 + \mathrm{DNR}_i)^p(1 + \mathrm{DNR}_j)^q[\mathbf{w}]_i^p[\mathbf{w}]_j^q \\ \text{and } [\mathbf{h}]_i^p[\mathbf{h}]_j^q &= \mathrm{DNR}_i^p \cdot \mathrm{DNR}_j^q \cdot [\mathbf{w}]_i^p[\mathbf{w}]_j^q. \end{aligned} \quad (32)$$

We note the true data of high-order monomial $[\bar{\mathbf{x}}]_i^p[\bar{\mathbf{x}}]_j^q$ is $[\mathbf{h}]_i^p[\mathbf{h}]_j^q$, the corresponding noise can thus be derived from the formula (32) as

$$\begin{aligned} &[\bar{\mathbf{x}}]_i^p[\bar{\mathbf{x}}]_j^q - [\mathbf{h}]_i^p[\mathbf{h}]_j^q \\ &\quad = \left[(1 + \mathrm{DNR}_i)^p(1 + \mathrm{DNR}_j)^q - \mathrm{DNR}_i^p \cdot \mathrm{DNR}_j^q\right][\mathbf{w}]_i^p[\mathbf{w}]_j^q \end{aligned}$$

which, in conjunction with the second formula in (32), leads to

$$\begin{aligned} \left|\mathrm{DNR}_{ij}^{p+q}\right| &\triangleq \left|\frac{[\mathbf{h}]_i^p[\mathbf{h}]_j^q}{[\bar{\mathbf{x}}]_i^p[\bar{\mathbf{x}}]_j^q - [\mathbf{h}]_i^p[\mathbf{h}]_j^q}\right| \\ &= \left|\frac{1}{\left(1 + \frac{1}{\mathrm{DNR}_i}\right)^p\left(1 + \frac{1}{\mathrm{DNR}_j}\right)^q - 1}\right|, \quad p, q \in \mathbb{N}. \end{aligned} \quad (33)$$

We can straightforwardly verify from formula (33) that if $\mathrm{DNR}_i,\ \mathrm{DNR}_j \in (0, \infty)$, we have

$$\left|\mathrm{DNR}_{ij}^{p+q}\right| = \frac{1}{\left(1 + \frac{1}{|\mathrm{DNR}_i|}\right)^p\left(1 + \frac{1}{|\mathrm{DNR}_j|}\right)^q - 1} \quad (34)$$

which implies $|\mathrm{DNR}_{ij}^{p+q}|$ is strictly increasing with respect to $|\mathrm{DNR}_i|$ and $|\mathrm{DNR}_j|$ under this condition.

The condition $\mathrm{DNR}_i,\ \mathrm{DNR}_j \in (-\infty, -1]$ means that

$$\begin{aligned} \frac{1}{\mathrm{DNR}_j} &\in [-1, 0), \quad \frac{1}{\mathrm{DNR}_j} \in [-1, 0) \\ 1 + \frac{1}{\mathrm{DNR}_i} &\in [0, 1), \quad 1 + \frac{1}{\mathrm{DNR}_j} \in [0, 1) \end{aligned} \quad (35)$$

by which, the formula (33) equivalently transforms to

$$\begin{aligned} \left|\mathrm{DNR}_{ij}^{p+q}\right| &= \frac{1}{1 - \left(1 + \frac{1}{\mathrm{DNR}_i}\right)^p\left(1 + \frac{1}{\mathrm{DNR}_j}\right)^q} \\ &= \frac{1}{1 - \left(1 - \frac{1}{|\mathrm{DNR}_i|}\right)^p\left(1 - \frac{1}{|\mathrm{DNR}_j|}\right)^q} \end{aligned} \quad (36)$$

which reveals that $|\mathrm{DNR}_{ij}^{p+q}|$ is strictly increasing with respect to $|\mathrm{DNR}_i|$ and $|\mathrm{DNR}_j|$.

The condition $\mathrm{DNR}_i,\ \mathrm{DNR}_j \in [-(1/2), 0)$ means

$$\begin{aligned} \frac{1}{\mathrm{DNR}_j} &\in (-\infty, -2], \quad \frac{1}{\mathrm{DNR}_j} \in (-\infty, -2] \\ 1 + \frac{1}{\mathrm{DNR}_i} &\in (-\infty, -1], \quad 1 + \frac{1}{\mathrm{DNR}_j} \in (-\infty, -1] \end{aligned} \quad (37)$$

in light of which, the formula (33) equivalently expresses as follows

1) If $p + q$ is even

$$\left|\mathrm{DNR}_{ij}^{m+n}\right| = \frac{1}{\left|\frac{1}{|\mathrm{DNR}_i|} - 1\right|^p\left|\frac{1}{|\mathrm{DNR}_j|} - 1\right|^q - 1}. \quad (38)$$

2) If $p + q$ is odd

$$\left|\mathrm{DNR}_{ij}^{p+q}\right| = \frac{1}{1 + \left|\frac{1}{|\mathrm{DNR}_i|} - 1\right|^p\left|\frac{1}{|\mathrm{DNR}_j|} - 1\right|^q}. \quad (39)$$

We note both the functions (38) and (39) imply $|\mathrm{DNR}_{ij}^{m+n}|$ is strictly increasing with respect to $|\mathrm{DNR}_i|$ and $|\mathrm{DNR}_j|$, which completes the proof. □

Theorem 5 [53]: For any pair of positive integers n and k, the number of n-tuples of nonnegative integers whose sum is r is equal to the number of multisets of cardinality $n - 1$ taken from a set of size $n + r - 1$, i.e., $\binom{n+r-1}{n-1} = (((n + r - 1)!)/((n - 1)!r!))$.

Theorem 6: The space complexity of Phy-Augmentation, i.e., the dimension of terminal output generated by Algorithm 1, in PhN layer (15), is

$$\mathrm{len}(\mathfrak{m}(\mathbf{x}, r)) = \sum_{s=1}^{r} \frac{(n + s - 1)!}{(n - 1)!s!} + 1. \quad (40)$$

Proof: We denote the output from Line 1 of Algorithm 1 by $\bar{\mathbf{x}}$. We first consider the case $r = 1$, in which Algorithm 1 skips Lines 4–15 and arrives at $\mathfrak{m}(\mathbf{x}, r) = [1;\ \bar{\mathbf{x}}]$ in Line 16. Noticing from Line 1 that $\bar{\mathbf{x}} \in \mathbb{R}^n$, we obtain $\mathrm{len}(\mathfrak{m}(\mathbf{x}, r)) = n + 1$, which verifies the correctness of (40) with $r = 1$.

We next consider the case $r \geq 2$. Given the input dimension n and order $s \in \{2, \ldots, r - 1, r\}$, the Lines 5–14 of Algorithm 1 are to generate all the nonmissing and nonredundant monomials included in $(\sum_{i=1}^{n}[\bar{\mathbf{x}}]_i)^s$. The problem of the number of generated monomials via Algorithm 1 is equivalent to the problem that for any pair of positive integers n and s, the number of n-tuples of nonnegative integers (whose sum is s) is equal to the number of multisets of cardinality $n-1$ taken from a set of size $n+s-1$. Additionally, we note that the vector generated in Line 13 of Algorithm 1, denoted by $\widetilde{\mathfrak{m}}(\mathbf{x}, s)$, stacks all the generated monomials. According to the auxiliary Theorem 5 in Appendix A, we then have $\mathrm{len}(\widetilde{\mathfrak{m}}(\mathbf{x}, s)) = (((n + s - 1)!)/((n - 1)!s!))$. Finally, we note that Lines 4,

and 16 of Algorithm 1 imply that the generated vector $\mathfrak{m}(\mathbf{x}, r)$ stack the 1 with $\widetilde{\mathfrak{m}}(\mathbf{x}, s)$ over $s \in \{1, \ldots, r-1, r\}$, respectively. We thus can obtain (40). □

Appendix B
Proof of Theorem 1

We note the $[\widetilde{\mathbf{h}}]_i$ given in (10) can be written as

$$[\widetilde{\mathbf{h}}]_i = \begin{cases} [\mathbf{h}]_i, & [\mathbf{h}]_i + [\mathbf{w}]_i < 0 \\ [\mathbf{h}]_i, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \text{ and } [\mathbf{w}]_i < 0 \\ [\mathbf{h}]_i \cdot \kappa_i + \rho_i, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \text{ and } [\mathbf{w}]_i > 0 \end{cases} \tag{41}$$

subtracting $[\mathbf{h}]_i$ from which yields

$$\left|[\widetilde{\mathbf{h}}]_i - [\mathbf{h}]_i\right| = \begin{cases} 0, & [\mathbf{h}]_i + [\mathbf{w}]_i < 0 \\ 0, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \text{ and } [\mathbf{w}]_i < 0 \\ |[\mathbf{h}]_i \cdot (\kappa_i - 1) + \rho_i|, & [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0 \text{ and } [\mathbf{w}]_i > 0. \end{cases} \tag{42}$$

Referring to the output $\chi([\bar{\mathbf{x}}]_i)$ of suppressor in (6), we can conclude that the $[\widetilde{\mathbf{h}}]_i$ given in (41) is the true data of suppressor output. Subtracting the $[\widetilde{\mathbf{h}}]_i$ from the $\chi([\bar{\mathbf{x}}]_i)$ results in the noise $[\widetilde{\mathbf{w}}]_i$ of suppressor output, given in (10).

We consider three cases to prove the property (8).

Case One: If $[\mathbf{h}]_i + [\mathbf{w}]_i < 0$, we obtain from the first item of $[\widetilde{\mathbf{h}}]_i$ in (41) and $[\widetilde{\mathbf{w}}]_i$ in (10) that $\text{DNR}_i = (([\widetilde{\mathbf{h}}]_i)/([\widetilde{\mathbf{w}}]_i)) = -1$.

Case Two: If $[\mathbf{h}]_i + [\mathbf{w}]_i \geq 0$ and $[\mathbf{w}]_i < 0$, we have $[\mathbf{h}]_i > 0$ and $[\mathbf{h}]_i > -[\mathbf{w}]_i| > 0$. We then obtain from the second item of $[\widetilde{\mathbf{h}}]_i$ in (41) and $[\widetilde{\mathbf{w}}]_i$ in (10) that $\text{DNR}_i = (([\widetilde{\mathbf{h}}]_i)/([\widetilde{\mathbf{w}}]_i)) = (([\mathbf{h}]_i)/([\mathbf{w}]_i)) < -1$.

Case Three: If $[\mathbf{h}]_i + [\mathbf{w}]_i \geq 0$ and $[\mathbf{w}]_i > 0$, we obtain from the third item of $[\mathbf{h}]_i$ in (41) and $[\widetilde{\mathbf{w}}]_i$ in (10) that $\text{DNR}_i = (([\widetilde{\mathbf{h}}]_i)/([\widetilde{\mathbf{w}}]_i)) = (([\mathbf{h}]_i \cdot \kappa_i + \rho_i)/([\mathbf{w}]_i \cdot \kappa_i))$. Recalling $[\widetilde{\mathbf{w}}]_i > 0$, if $\kappa_i > 0$, the $(([\mathbf{h}]_i \cdot \kappa_i + \rho_i)/([\mathbf{w}]_i \cdot \kappa_i)) \leq -1$ is equivalent to

$$\rho_i \leq -([\mathbf{h}]_i + [\mathbf{w}]_i)\kappa_i < 0, \quad \text{with } \kappa_i > 0, \quad [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0. \tag{43}$$

If $\kappa_i < 0$, the $(([\mathbf{h}]_i \cdot \kappa_i + \rho_i)/([\mathbf{w}]_i \cdot \kappa_i)) \leq -1$ is equivalent to

$$\rho_i \geq -([\mathbf{w}]_i + [\mathbf{h}]_i)\kappa_i \geq 0, \quad \text{with } \kappa_i < 0, \quad [\mathbf{h}]_i + [\mathbf{w}]_i \geq 0. \tag{44}$$

We finally conclude from (43) and (44) that $\text{DNR}_i = (([\widetilde{\mathbf{h}}]_i)/([\widetilde{\mathbf{w}}]_i)) \in (-\infty, -1]$ under condition (7). According to Theorem 4, we arrive at property (8), which completes the proof.

Appendix C
Proof of Theorem 2

We first consider the case $t = 1$. Line 5 of Algorithm 2 means that the knowledge matrix $\mathbf{K}_{\langle 1\rangle}$ includes all the elements of knowledge set $\mathbb{K}$, whose corresponding entries in the masking matrix $\mathbf{M}_{\langle 1\rangle}$ are frozen to be zeros. Consequently, both $\mathbf{M}_{\langle 1\rangle} \odot \mathbf{A}$ and $\mathbf{U}_{\langle 1\rangle} = \mathbf{M}_{\langle 1\rangle} \odot \mathbf{W}_{\langle 1\rangle}$ exclude all the elements of knowledge set $\mathbb{K}$. With the consideration of 2, we thus conclude that $\mathbf{M}_{\langle 1\rangle} \odot \mathbf{A} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}) + \mathbf{f}(\mathbf{x})$ in the ground-truth model (11) and $\mathbf{a}_{\langle 1\rangle} \odot \text{act}(\mathbf{U}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}))$ in the output computation (12) are independent of the term $\mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle})$. Moreover, the activation-masking vector indicates that the activation function corresponding to the output's ith entry is inactive if all the entries in the ith row of the masking matrix are zeros. We arrive at the conclusion that the first PhN layer strictly complies with the available physical knowledge about the ground truth (11), i.e., if the $[\mathbf{A}]_{i,j} \in \mathbb{K}$, the $((\partial[\mathbf{y}_{\langle 1\rangle}]_i)/(\partial[\mathfrak{m}(\mathbf{x}, r)]_j)) \equiv ((\partial[\mathbf{y}]_i)/(\partial[\mathfrak{m}(\mathbf{x}, r)]_j)) \equiv [\mathbf{A}]_{i,j}$ always holds.

We next consider the remaining PhN layers. Considering the Line 16 Algorithm 2, we have

$$\begin{aligned}
&[\mathbf{y}_{\langle p\rangle}]_{1:\text{len}(\mathbf{y})} \\
&= [\mathbf{K}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})]_{1:\text{len}(\mathbf{y})} \\
&\quad + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= \mathbf{I}_{\text{len}(\mathbf{y})} \cdot [\mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})]_{2:(\text{len}(\mathbf{y})+1)} \\
&\quad + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \qquad (45a) \\
&= \mathbf{I}_{\text{len}(\mathbf{y})} \cdot [\mathbf{y}_{\langle p-1\rangle}]_{1:\text{len}(\mathbf{y})} \\
&\quad + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \qquad (45b) \\
&= [\mathbf{y}_{\langle p-1\rangle}]_{1:\text{len}(\mathbf{y})} + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= [\mathbf{K}_{\langle p-1\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-2\rangle}, r_{\langle p-1\rangle})]_{1:\text{len}(\mathbf{y})} \\
&\quad + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= \mathbf{I}_{\text{len}(\mathbf{y})} \cdot [\mathfrak{m}(\mathbf{y}_{\langle p-2\rangle}, r_{\langle p-1\rangle})]_{2:(\text{len}(\mathbf{y})+1)} \\
&\quad + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= \mathbf{I}_{\text{len}(\mathbf{y})} \cdot [\mathbf{y}_{\langle p-2\rangle}]_{1:\text{len}(\mathbf{y})} \\
&\quad + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= [\mathbf{y}_{\langle p-2\rangle}]_{1:\text{len}(\mathbf{y})} + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \breve{\mathfrak{m}}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= \cdots = [\mathbf{y}_{\langle 1\rangle}]_{1:\text{len}(\mathbf{y})} + [\mathbf{a}_{\langle p\rangle} \odot \text{act}(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle}))]_{1:\text{len}(\mathbf{y})} \\
&= [\mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle})]_{1:\text{len}(\mathbf{y})} \\
&\quad + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= \mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}) + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \qquad (45c)
\end{aligned}$$

where (45a) and (45b) are obtained from their previous steps via considering the structure of block matrix $\mathbf{K}_{\langle t\rangle}$ (generated in Line 10 of Algorithm 2) and the formula of augmented monomials: $\mathfrak{m}(\mathbf{x}, r) = [1;\ \mathbf{x};\ [\mathfrak{m}(\mathbf{x}, r)]_{(\text{len}(\mathbf{x})+2):\text{len}(\mathfrak{m}(\mathbf{x},r))}]$ (generated via Algorithm 1). The remaining iterative steps follow the same path.

The training loss function is to push the terminal output of Algorithm 2 (i.e., $\widehat{\mathbf{y}} = \mathbf{y}_{\langle p\rangle}$) to approximate the real output $\mathbf{y}$, which in light of (45c) yields

$$\begin{aligned}
&\widehat{\mathbf{y}} \\
&= \mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}) + [\mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big)]_{1:\text{len}(\mathbf{y})} \\
&= \mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle}) + \mathbf{a}_{\langle p\rangle} \odot \text{act}\big(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle})\big) \qquad (46)
\end{aligned}$$

where (46) from its previous step is obtained via considering the fact $\text{len}(\widehat{\mathbf{y}}) = \text{len}(\mathbf{y}) = \text{len}(\mathbf{y}_{\langle p\rangle})$. Meanwhile, the condition of generating a weight-masking matrix in Line 11 of Algorithm 2 removes all the nodal connections with the elements of knowledge set $\mathbb{K}$ included in $\mathbf{K}_{\langle 1\rangle}$. Therefore, we can conclude that in the terminal output computation (46), the term $\mathbf{a}_{\langle p\rangle} \odot \text{act}(\mathbf{U}_{\langle p\rangle} \cdot \mathfrak{m}(\mathbf{y}_{\langle p-1\rangle}, r_{\langle p\rangle}))$ does not have an influence on the computing of knowledge term $\mathbf{K}_{\langle 1\rangle} \cdot \mathfrak{m}(\mathbf{x}, r_{\langle 1\rangle})$. Thus, the Algorithm 2 strictly embeds and preserves the available knowledge about the physics model of ground truth (1), or equivalently the (11).

APPENDIX D
PROOF OF THEOREM 3

By Theorem 6 in Appendix A, the number of augmented monomials of d cascading PhNs (16) is obtained as

$$\sum_{p=1}^{d}\text{len}(\mathfrak{m}(\mathbf{x},r_{\langle p\rangle}))=\underbrace{\sum_{s=1}^{r_{\langle 1\rangle}}\frac{(n+s-1)!}{(n-1)s!}+1}_{\text{the first PhN}}+\underbrace{\sum_{v=1}^{d-1}\sum_{s=1}^{r_{\langle v+1\rangle}}\frac{(n_{\langle v\rangle}+s-1)!}{(n_{\langle v\rangle}-1)!s!}+d-1}_{\text{the remaining PhNs}}. \tag{47}$$

The condition (17) implies that $r>r_{\langle 1\rangle}$, which in conjunction with (40), lead to

$$\text{len}(\mathfrak{m}(\mathbf{x},r))=\sum_{s=1}^{r_{\langle 1\rangle}}\frac{(n+s-1)!}{(n-1)!s!}+\sum_{s=r_{\langle 1\rangle}+1}^{r}\frac{(n+s-1)!}{(n-1)!s!}+1. \tag{48}$$

Subtracting (47) from (48) yields (18).

APPENDIX E
DERIVATIONS OF SOLUTION (27)

With the consideration of (23)–(26), the safety formulas: $[\mathbf{s}(\mathbf{u}(k))]_1=[\widehat{\mathbf{c}}]_1$ and $[\mathbf{s}(\mathbf{u}(k))]_2=[\widehat{\mathbf{c}}]_2$ can be rewritten as

$$\lambda_1[\widehat{\mathbf{u}}(k)]_1^2+\lambda_2[\widehat{\mathbf{u}}(k)]_2^2=[\widehat{\mathbf{c}}]_1-[\mathbf{b}]_1 \tag{49}$$

$$\begin{bmatrix}\theta(k)\\ \gamma(k)\end{bmatrix}^{\top}\mathbf{P}_2\begin{bmatrix}\theta(k)\\ \gamma(k)\end{bmatrix}=s_{11}[\widehat{\mathbf{u}}(k)]_1^2+2s_{12}[\widehat{\mathbf{u}}(k)]_1[\widehat{\mathbf{u}}(k)]_2+s_{22}[\widehat{\mathbf{u}}(k)]_2^2=[\mathbf{b}]_2-[\widehat{\mathbf{c}}]_2. \tag{50}$$

We now define

$$\overline{\mu}_1\triangleq\frac{[\widehat{\mathbf{c}}]_1-[\mathbf{b}]_1}{\lambda_1},\qquad\overline{\lambda}\triangleq\frac{\lambda_2}{\lambda_1},\qquad\bar{b}\triangleq[\mathbf{b}]_2-[\widehat{\mathbf{c}}]_2 \tag{51}$$

leveraging which, the (49) is rewritten as

$$[\widehat{\mathbf{u}}(k)]_1^2=\overline{\mu}_1-\overline{\lambda}[\widehat{\mathbf{u}}(k)]_2^2 \tag{52}$$

and we can obtain from (50) that

$$\begin{aligned}4s_{12}^2[\widehat{\mathbf{u}}(k)]_2^2[\widehat{\mathbf{u}}(k)]_1^2&=\bar{b}^2+s_{11}^2[\widehat{\mathbf{u}}(k)]_1^4+s_{22}^2[\widehat{\mathbf{u}}(k)]_2^4-2\bar{b}s_{11}[\widehat{\mathbf{u}}(k)]_1^2\\&\quad-2\bar{b}s_{22}[\widehat{\mathbf{u}}(k)]_2^2+2s_{11}s_{22}[\widehat{\mathbf{u}}(k)]_1^2[\widehat{\mathbf{u}}(k)]_2^2\end{aligned}$$

substituting (52) into which yields

$$\varpi_1[\widehat{\mathbf{u}}]_2^4(k)+\varpi_2[\widehat{\mathbf{u}}]_2^2(k)+\varpi_3(k)=0 \tag{53}$$

where

$$\varpi_1\triangleq s_{11}^2\overline{\lambda}^2+s_{22}^2-2s_{11}s_{22}\overline{\lambda}+4s_{12}^2\overline{\lambda} \tag{54}$$

$$\varpi_2\triangleq 2\bar{b}s_{11}\overline{\lambda}-2s_{11}^2\overline{\mu}_1\overline{\lambda}-2\bar{b}s_{22}+2s_{11}s_{22}\overline{\mu}_1-4s_{12}^2\overline{\mu}_1 \tag{55}$$

$$\varpi_3\triangleq\bar{b}^2+s_{11}^2\overline{\mu}_1^2-2\bar{b}s_{11}\overline{\mu}_1. \tag{56}$$

Considering $\widehat{u}_2^2(k)\geq 0$, the solution of (53) is

$$[\widehat{\mathbf{u}}]_2^2(k)=\frac{\sqrt{\varpi_2^2-4\varpi_1\varpi_3}-\varpi_2}{2\varpi_1} \tag{57}$$

substituting which into (52) yields

$$[\widehat{\mathbf{u}}]_1^2(k)=\overline{\mu}_1-\overline{\lambda}\frac{\sqrt{\varpi_2^2-4\varpi_1\varpi_3}-\varpi_2}{2\varpi_1}. \tag{58}$$

The $\widehat{\mathbf{u}}(k)$ is straightforwardly obtained from (57) and (58)

$$\widehat{\mathbf{u}}(k)=\left[\pm\sqrt{\overline{\mu}_1-\overline{\lambda}\frac{\sqrt{\varpi_2^2-4\varpi_1\varpi_3}-\varpi_2}{2\varpi_1}};\ \pm\sqrt{\frac{\sqrt{\varpi_2^2-4\varpi_1\varpi_3}-\varpi_2}{2\varpi_1}}\right]$$

which, in conjunction with (26) and $Q^{-1}=Q=Q^{\top}$, leads to

$$\begin{bmatrix}\theta(k)\\ \gamma(k)\end{bmatrix}=\mathbf{Q}_1\begin{bmatrix}\pm\sqrt{\overline{\mu}_1-\overline{\lambda}\frac{\sqrt{\varpi_2^2-4\varpi_1\varpi_3}-\varpi_2}{2\varpi_1}}\\ \pm\sqrt{\frac{\sqrt{\varpi_2^2-4\varpi_1\varpi_3}-\varpi_2}{2\varpi_1}}\end{bmatrix}. \tag{59}$$

Substituting the notations defined in (51) into (59) and (54)–(56) results in (27) and (28)–(30), respectively.

REFERENCES

[1] A. Zachary and T. Helen, "AI accidents: An emerging threat," Center Secur. Emerg. Technol., Washington, DC, USA, Tech. Rep., 2021, doi: 10.51593/20200072.

[2] Z. Fang, "A high-efficient hybrid physics-informed neural networks based on convolutional neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5514–5526, Oct. 2022.

[3] R. Wang and R. Yu, "Physics-guided deep learning for dynamical systems: A survey," 2021, *arXiv:2107.01272*.

[4] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar, "Integrating scientific knowledge with machine learning for engineering and environmental systems," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–37, Apr. 2023.

[5] X. Jia et al., "Physics-guided machine learning for scientific discovery: An application in simulating lake temperature profiles," *ACM/IMS Trans. Data Sci.*, vol. 2, no. 3, pp. 1–26, Aug. 2021.

[6] X. Jia et al., "Physics-guided RNNs for modeling dynamical systems: A case study in simulating lake temperature profiles," in *Proc. SIAM Int. Conf. Data Mining*, 2019, pp. 558–566.

[7] S. Wang and P. Perdikaris, "Deep learning of free boundary and Stefan problems," *J. Comput. Phys.*, vol. 428, Mar. 2021, Art. no. 109914.

[8] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, and S. G. Johnson, "Physics-informed neural networks with hard constraints for inverse design," *SIAM J. Sci. Comput.*, vol. 43, no. 6, pp. B1105–B1132, Jan. 2021.

[9] Y. Chen et al., "Theory-guided hard constraint projection (HCP): A knowledge-based data-driven scientific machine learning method," *J. Comput. Phys.*, vol. 445, Nov. 2021, Art. no. 110624.

[10] N. Wang, D. Zhang, H. Chang, and H. Li, "Deep learning of subsurface flow via theory-guided neural network," *J. Hydrol.*, vol. 584, May 2020, Art. no. 124700.

[11] K. Xu and E. Darve, "Physics constrained learning for data-driven inverse modeling from sparse observations," *J. Comput. Phys.*, vol. 453, Mar. 2022, Art. no. 110938.

[12] G. E. Karniadakis et al., "Physics-informed machine learning," *Nat. Rev. Phys.*, vol. 3, no. 6, pp. 422–440, 2021.

[13] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 1457–1466.

[14] A. Daw, A. Karpatne, W. Watkins, J. Read, and V. Kumar, "Physics-guided neural networks (PGNN): An application in lake temperature modeling," 2017, *arXiv:1710.11431*.

[15] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho, "Lagrangian neural networks," in *Proc. ICLR Workshop Integr. Deep Neural Models Differ. Equ.*, 2020, pp. 1–9.

[16] M. Finzi, K. A. Wang, and A. G. Wilson, "Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 13880–13889.

[17] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[18] M. Z. Yousif, L. Yu, and H. Lim, "Physics-guided deep learning for generating turbulent inflow conditions," *J. Fluid Mech.*, vol. 936, p. 21, Apr. 2022.

[19] W. Li, M. Z. Bazant, and J. Zhu, "A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches," *Comput. Methods Appl. Mech. Eng.*, vol. 383, Sep. 2021, Art. no. 113933.

[20] C. Rao, H. Sun, and Y. Liu, "Hard encoding of physics for learning spatiotemporal dynamics," 2021, *arXiv:2105.00557*.

[21] N. Muralidhar et al., "PhyNet: Physics guided neural networks for particle drag force prediction in assembly," in *Proc. SIAM Int. Conf. Data Mining*, 2020, pp. 559–567.

[22] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst, "Geodesic convolutional neural networks on Riemannian manifolds," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 832–840.

[23] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5425–5434.

[24] M. Horie, N. Morita, T. Hishinuma, Y. Ihara, and N. Mitsume, "Isometric transformation invariant and equivariant graph convolutional networks," 2020, *arXiv:2005.06316*.

[25] R. Wang, "Incorporating symmetry into deep dynamics models for improved generalization," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–20.

[26] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba, "Learning compositional Koopman operators for model-based control," 2019, *arXiv:1910.08264*.

[27] S. Wang, H. Wang, and P. Perdikaris, "Learning the solution operator of parametric partial differential equations with physics-informed DeepONets," *Sci. Adv.*, vol. 7, no. 40, Oct. 2021, Art. no. eabi8605.

[28] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Commun.*, vol. 9, no. 1, pp. 1–10, Nov. 2018.

[29] S. A. Faroughi et al., "Physics-guided, physics-informed, and physics-encoded neural networks in scientific computing," 2022, *arXiv:2211.07377*.

[30] I. E. Lagaris, A. Likas, and D. I. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 987–1000, Mar. 1998.

[31] Y.-Y. Yang, C.-N. Chou, and K. Chaudhuri, "Understanding rare spurious correlations in neural networks," 2022, *arXiv:2202.05189*.

[32] S. Sagawa, A. Raghunathan, P. W. Koh, and P. Liang, "An investigation of why overparameterization exacerbates spurious correlations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8346–8356.

[33] J. Nagoor Kani and A. H. Elsheikh, "DR-RNN: A deep residual recurrent neural network for model reduction," 2017, *arXiv:1709.00939*.

[34] F. D. A. Belbute-Peres, T. Economon, and Z. Kolter, "Combining differentiable PDE solvers and graph neural networks for fluid flow prediction," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 2402–2411.

[35] D. Wu et al., "DeepGLEAM: A hybrid mechanistic and deep learning model for COVID-19 forecasting," 2021, *arXiv:2102.06684*.

[36] V. Le Guen and N. Thome, "Disentangling physical dynamics from unknown factors for unsupervised video prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11471–11481.

[37] V. Garcia Satorras, Z. Akata, and M. Welling, "Combining generative and discriminative models for hybrid inference," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1–11.

[38] Y. Long, X. She, and S. Mukhopadhyay, "HybridNet: Integrating model-based and data-driven learning to predict evolution of dynamical systems," in *Proc. Conf. Robot Learn.*, 2018, pp. 551–560.

[39] Y. Yin et al., "Augmenting physical models with deep networks for complex dynamics forecasting*," *J. Stat. Mech., Theory Exp.*, vol. 2021, no. 12, Dec. 2021, Art. no. 124012.

[40] H. Cao, Y. Mao, L. Sha, and M. Caccamo, "Physical deep reinforcement learning towards safety guarantee," 2023, *arXiv:2303.16860*.

[41] K. Königsberger, *Analysis 2*. Cham, Switzerland: Springer-Verlag, 2013.

[42] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 3420–3431.

[43] A. Singletary, A. Swann, Y. Chen, and A. D. Ames, "Onboard safety guarantees for racing drones: High-speed geofencing with control barrier functions," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2897–2904, Apr. 2022.

[44] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Trans. Robot.*, vol. 34, no. 6, pp. 1603–1622, Dec. 2018.

[45] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, May 2007.

[46] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 3882–3889.

[47] J. C. Doyle, B. A. Francis, and A. R. Tannenbaum, *Feedback Control Theory*. North Chelmsford, MA, USA: Courier Corporation, 2013.

[48] B. Goldfain et al., "AutoRally: An open platform for aggressive autonomous driving," *IEEE Control Syst. Mag.*, vol. 39, no. 1, pp. 26–55, Feb. 2019.

[49] Y. Mao, L. Sha, H. Shao, Y. Gu, Q. Wang, and T. Abdelzaher, "Phy-Taylor: Physics-model-based deep neural networks," 2022, *arXiv:2209.13511*.

[50] R. Rajamani, *Vehicle Dynamics and Control*. Cham, Switzerland: Springer, 2011.

[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[52] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*.

[53] R. P. Stanley, "What is enumerative combinatorics?" in *Enumerative Combinatorics*. Cham, Switzerland: Springer, 1986, pp. 1–63.