

# Domain-Decoupled Physics-informed Neural Networks with Closed-Form Gradients for Fast Model Learning of Dynamical Systems

Henrik Krauss<sup>1,2</sup><sup>a</sup>, Tim-Lukas Habich<sup>2</sup><sup>b</sup>, Max Bartholdt<sup>2</sup><sup>c</sup>,  
Thomas Seel<sup>2</sup><sup>d</sup> and Moritz Schappler<sup>2</sup><sup>e</sup>

<sup>1</sup>Department of Advanced Interdisciplinary Studies, The University of Tokyo, Tokyo, Japan

<sup>2</sup>Institute of Mechatronic Systems, Leibniz University Hannover, 30823 Garbsen, Germany

{tim-lukas.habich, max.bartholdt, thomas.seel, moritz.schappler}@imes.uni-hannover.de

**Keywords:** Physics-Informed Machine Learning, Surrogate Modeling, Model Learning, System Dynamics.


**Abstract:** Physics-informed neural networks (PINNs) are trained using physical equations and can also incorporate unmodeled effects by learning from data. PINNs for control (PINC) of dynamical systems are gaining interest due to their prediction speed compared to classical numerical integration methods for nonlinear state-space models, making them suitable for real-time control applications. We introduce the domain-decoupled physics-informed neural network (DD-PINN) to address current limitations of PINC in handling large and complex nonlinear dynamical systems. The time domain is decoupled from the feed-forward neural network to construct an Ansatz function, allowing for calculation of gradients in closed form. This approach significantly reduces training times, especially for large dynamical systems, compared to PINC, which relies on graph-based automatic differentiation. Additionally, the DD-PINN inherently fulfills the initial condition and supports higher-order excitation inputs, simplifying the training process and enabling improved prediction accuracy. Validation on three systems – a nonlinear mass-spring-damper, a five-mass-chain, and a two-link robot – demonstrates that the DD-PINN achieves significantly shorter training times. In cases where the PINC’s prediction diverges, the DD-PINN’s prediction remains stable and accurate due to higher physics loss reduction or use of a higher-order excitation input. The DD-PINN allows for fast and accurate learning of large dynamical systems previously out of reach for the PINC.


## 1 INTRODUCTION


Physics-informed neural networks (PINNs) have been introduced by (Raissi et al., 2019) as a machine-learning framework to solve ordinary (ODEs) and partial differential equations (PDEs). On top of supervised learning of a feed-forward neural network (FNN) on data, PINNs introduce a custom physics loss based on the system-governing ODEs or PDEs including boundary/initial conditions. This seamless integration of physical laws into the neural-network training process leads to *accurate predictions even with limited data* as well as the ability to *extrapolate on out-of-distribution (unseen) data*, while serving as a *fast surrogate model*. So far, many variants


of PINNs have been successfully applied to fields such as thermodynamics, chemistry, material science (Karniadakis et al., 2021; Cuomo et al., 2022), and dynamical systems, including robotics (Hao et al., 2022). PINNs have been adapted for the control of dynamical systems (PINC) by (Antonelo et al., 2024) where the FNN predicts the system’s evolution over a short time interval assuming constant excitation. This concept is compatible with and has been applied to state estimation using an Unscented Kalman Filter (UKF) (de Curtò and de Zarzà, 2024), model predictive control (MPC) of robotic systems (Nghiem et al., 2023), such as two-link manipulators (Nicode-mus et al., 2022; Yang et al., 2023) and quadrotors (Sanyal and Roy, 2023). Here, the PINC outperforms traditional numerical integrators regarding prediction speed of the system dynamics.


However, the training of PINNs can be challenging and many efforts to improve trainability have been developed (Wang et al., 2023). These include adap-

<sup>a</sup> <https://orcid.org/0000-0002-9787-5465>

<sup>b</sup> <https://orcid.org/0000-0003-4167-8443>

<sup>c</sup> <https://orcid.org/0000-0002-8422-9368>

<sup>d</sup> <https://orcid.org/0000-0002-6920-1690>

<sup>e</sup> <https://orcid.org/0000-0001-7952-7363>

tive activation functions (Jagtap et al., 2020), various loss-balancing techniques such as GradNorm (Chen et al., 2018), SoftAdapt (Heydari et al., 2019), learning rate-annealing (LRA) (Wang et al., 2021), relative loss balancing with random lookback (ReLoBRaLo) (Bischof and Kraus, 2021), dynamically normalized PINNs (DN-PINNs) (Deguchi and Asai, 2023), adaptive collocation point sampling strategies (Wu et al., 2023), as well as training on non-dimensional PDEs (Kapoor et al., 2024). Also, for dynamical systems with chaotic behaviors, loss functions might need to be reformulated to respect spatio-temporal causality to ensure training convergence (Wang et al., 2022).

One main point of research for PINNs is the topic of differentiation of the FNN output with respect to the spatial and temporal *domain* inputs, i.e., the independent variables in the ODE or PDE over which the solution is defined. Differentiation is required to evaluate the ODE/PDE and determine the physics loss but is *very computationally expensive*. Training times of the PINC (Antonelo et al., 2024) can amount to several days or weeks, *rendering the learning of large or complex dynamical systems unfeasible*. Here, previously developed architectures can be summarized in the five categories of using

- (i) automatic differentiation (AD),
- (ii) numeric differentiation,
- (iii) a hybrid of (i)–(ii),
- (iv) a variational approach, or
- (v) closed-form gradients.

(i) Classical PINNs as formulated by (Raissi et al., 2019) or (Berg and Nyström, 2018) use graph-based *automatic differentiation* (a-PINNs) as implemented in PyTorch or TensorFlow to determine the FNN gradients. However, AD is computationally expensive and a-PINNs can only reach high accuracy for large numbers of collocation points. (ii) PINNs based on *numerical differentiation* (n-PINNs) avoid AD by using numerical differentiation on the collocation points. This makes them more robust to a low number of collocation points but may attain lower accuracy than a-PINNs (Cuomo et al., 2022). (iii) Addressing the disadvantages of both approaches, (Chiu et al., 2022) have used a coupled-automatic-numerical (CAN) differentiation method and introduced *CAN-PINNs* – a hybrid approach between a- and n-PINNs. Their residual loss is formulated at locally adjacent collocation points through a Taylor-series scheme that includes derivatives at the collocation points obtained from automatic differentiation. However, they require the selection of an appropriate numerical scheme and therefore cannot be straightforwardly interchanged with a-PINNs. (iv) Another approach are *variational*

*PINNs* (v-PINNs) introduced in (Kharazmi et al., 2019). Here, a variational loss function is constructed within the Petrov-Galerkin framework. For that, the PINNs' output is numerically integrated and tested in a test space of Legendre polynomials. Variants of the v-PINN have been developed that divide the domain into subdomains (Kharazmi et al., 2021; Liu and Wu, 2023) (v) Also, it is possible to compute *analytical gradients* without graph-based AD, such as for single hidden-layer networks (Lagaris et al., 1998). Alternatively, the FNN does not predict the solution to the ODE or PDE directly but parameters of basis or Ansatz functions that approximate the solution. Examples using this approach are:

- Taylor PINN (Zhang et al., 2023),
- Polynomial-interpolation PINN (Tang et al., 2023),
- Legendre-improved extreme learning machines (Yang et al., 2020), and
- Multiwavelet-based neural operators (Gupta et al., 2021).

In the context of PINNs for control, to the best of our knowledge, only AD-based architectures such as the PINC have been applied (Antonelo et al., 2024), featuring the assumption of constant excitation in the short prediction horizon. Three main limitations of this PINC for the learning of large and nonlinear dynamical systems are:

- (I) Graph-based automatic differentiation causes *long training times*. The number of AD operations  $n_{AD}$  per epoch is proportional to the number of system states  $m$  and the number of collocation points  $n_{collo}$ , where  $n_{collo}$  itself may be chosen higher for larger or more complex systems.
- (II) Constant excitation (zero-order-hold assumption) for longer trained time intervals results in *poor accuracy*. However, longer time intervals are advantageous for fast simulation since one prediction with the surrogate model should simulate a large time step with high accuracy.
- (III) Initial-condition (IC) loss and physics loss can *require a weighting scheme* such as GradNorm, SoftAdapt, LRA or ReLoBRaLo. These weighting schemes need to be chosen empirically, are sensitive to their hyper-parameter settings, and can further increase training time.

For the prediction horizons used in control scenarios, approximating the solution through differentiable Ansatz functions to calculate gradients in closed-form has the potential to drastically reduce training times while maintaining high accuracy. For the first time,

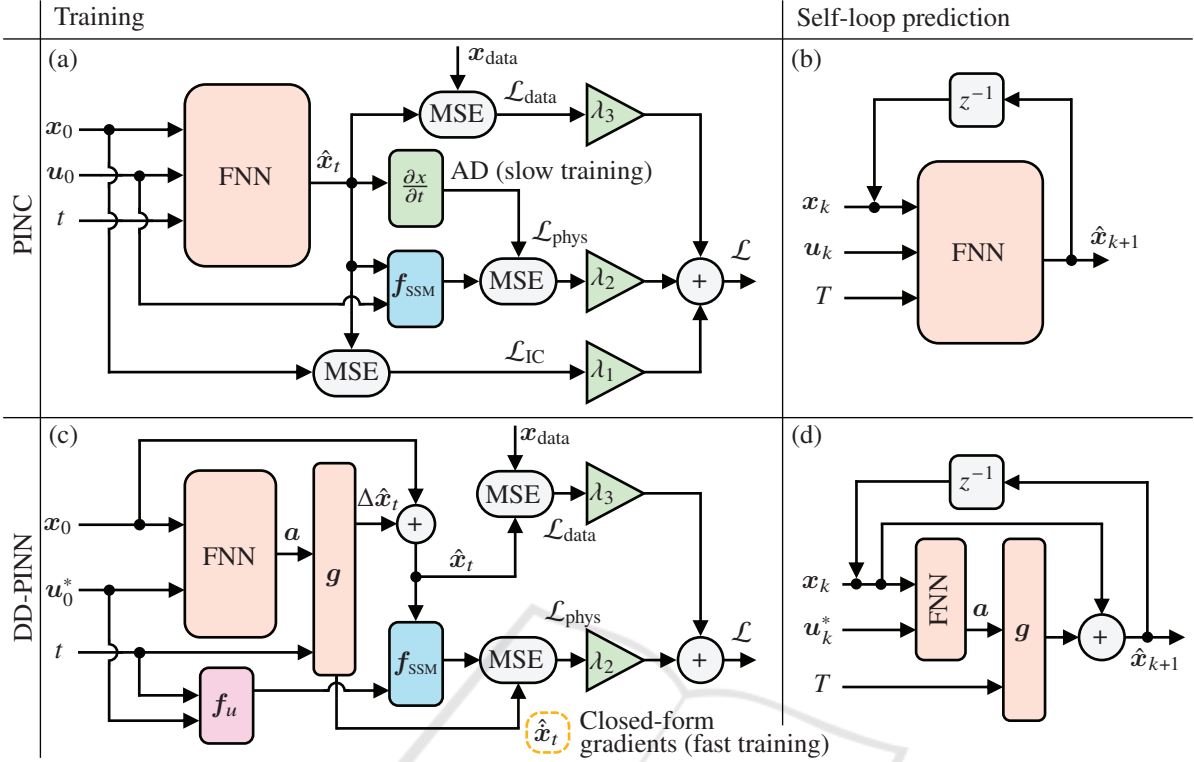


Figure 1: Configurations for (a) PINN training and (b) PINN self-loop prediction process as well as (c) DD-PINN training and (d) DD-PINN self-loop prediction. Time domain is decoupled from the FNN for the DD-PINN, enabling calculation of gradients in closed form.

we apply the approach (v) of using analytical, closed-form gradients for state-space learning of dynamical systems. We propose a domain-decoupled PINN (DD-PINN) architecture where the time domain is decoupled from the FNN to construct an Ansatz function that enables calculation of gradients in closed form. It is formulated without being limited to a specific Ansatz function, unlike previous work, and can be employed interchangeably with the PINN for physics-loss calculation during training as a state estimator or in control such as for model predictive control. The DD-PINN is evaluated on three exemplary dynamical systems with varying system complexity and compared to the PINN (Nicodemus et al., 2022; Antonelo et al., 2024) with regards to its limitations.

In Sec. 2, the fundamentals of the PINN are explained and the DD-PINN is formulated in Sec. 3. The validation of the DD-PINN and a comparison to the PINN follows in Sec. 4, preceding the conclusion in chapter 5.

## 2 PRELIMINARIES

In the original PINN for control (PINN) (Antonelo et al., 2024), dynamical systems in form of a state-space model (SSM)

$$\dot{\mathbf{x}}(t) = \mathbf{f}_{\text{SSM}}(\mathbf{x}(t), \mathbf{u}(t)) \quad (1)$$

are approximated using a neural network, where  $\mathbf{x}$  denotes the state vector,  $t$  the time domain and  $\mathbf{u}$  an input signal. The PINN predicts the future state

$$\hat{\mathbf{x}}_t = \mathbf{f}_{\text{NN}}(\mathbf{x}_0, \mathbf{u}_0, t, \theta) \approx \mathbf{x}(t) \quad (2)$$

with the feed-forward neural network  $\mathbf{f}_{\text{NN}}$  continuously over the sampling interval

$$0 \leq t \leq T_s \quad (3)$$

in which the excitation is assumed constant and subject to boundaries

$$\begin{aligned} \mathbf{x}_{\min} &\leq \mathbf{x}_0 \leq \mathbf{x}_{\max}, \\ \mathbf{u}_{\min} &\leq \mathbf{u}_0 \leq \mathbf{u}_{\max}. \end{aligned} \quad (4)$$

The parameters, i.e., the weights and biases  $\theta$  of the FNN are trained according to the process visualized in Fig. 1(a) using a custom physics-informed loss function

$$\mathcal{L}(\theta) = \lambda_1 \mathcal{L}_{\text{IC}}(\theta) + \lambda_2 \mathcal{L}_{\text{phys}}(\theta) + \lambda_3 \mathcal{L}_{\text{data}}(\theta) \quad (5)$$

consisting of the initial-condition loss

$$\mathcal{L}_{\text{IC}} = \text{MSE}(\mathbf{x}_0, \hat{\mathbf{x}}_0), \quad (6)$$

physics loss

$$\mathcal{L}_{\text{phys}} = \text{MSE}\left(\frac{\partial}{\partial t} \hat{\mathbf{x}}_t, \mathbf{f}_{\text{SSM}}(\hat{\mathbf{x}}_t, \mathbf{u}_0)\right), \quad (7)$$

and data loss

$$\mathcal{L}_{\text{data}} = \text{MSE}(\hat{\mathbf{x}}_t, \mathbf{x}_{\text{data}}). \quad (8)$$

The mean squared error is denoted by  $\text{MSE}()$ . The initial-condition loss, if sufficiently minimized, ensures that for  $t = 0$  the PINC predicts the initial state  $\mathbf{x}_0$ . The physics loss  $\mathcal{L}_{\text{phys}}$  includes the system's dynamics equation in state-space form and therefore governs learning of the physics model. The derivative of the predicted state  $\frac{\partial}{\partial t} \hat{\mathbf{x}}_t$  is calculated through graph-based automatic differentiation with respect to the time input  $t$ . In principle, these two losses are sufficient to learn the system dynamics. Optionally, a data loss  $\mathcal{L}_{\text{data}}$  can be added, to include data sets obtained from real hardware to incorporate effects that may not be captured by the first-principles model. As these losses generally have different magnitudes and convergence behaviors, they are weighted using the factors  $\lambda_i$ , which are either determined empirically or by loss-weighting schemes as mentioned in Sec. 1.

In the training process, the loss functions  $\mathcal{L}_{\text{IC}}$  and  $\mathcal{L}_{\text{phys}}$  are evaluated on randomly sampled collocation points in the intervals given in (3) and (4). One sampling method commonly applied is Latin hypercube sampling. Using backpropagation, the FNN parameters  $\theta$  are optimized using the total loss in (5).

After training, the PINC can be employed for dynamic state prediction in self loop as visualized in Fig. 1(b) where  $k$  indicates the current step. The step size  $T < T_s$  is chosen constant for numerical integration at an operation frequency  $f = \frac{1}{T}$ .

### 3 DOMAIN-DECOUPLED PINN

We propose an alternative architecture to the PINC, called the domain-decoupled physics-informed neural network (DD-PINN) to address the limitations listed in Sec. 1. The architecture of the DD-PINN in comparison to the PINC is visualized in Fig. 1(c-d) for the training and prediction process, respectively. The main modification involves decoupling the time domain via

$$\begin{aligned} \hat{\mathbf{x}}_t &= \mathbf{g}(\mathbf{f}_{\text{NN}}(\mathbf{x}_0, \mathbf{u}_0, \theta), t) + \mathbf{x}_0 \\ &= \mathbf{g}(\mathbf{a}, t) + \mathbf{x}_0 \end{aligned} \quad (9)$$

where  $\mathbf{f}_{\text{NN}}$  now predicts the vector

$$\mathbf{a} = \mathbf{f}_{\text{NN}}(\mathbf{x}_0, \mathbf{u}_0, \theta) \quad (10)$$

which is used in an Ansatz function  $\mathbf{g}$ . We restrict  $\mathbf{g}$  to the following two properties:

1.  $\mathbf{g}$  is differentiable in closed form  $\forall t \in [0, T_s]$
2.  $\mathbf{g}(\mathbf{a}, 0) \equiv \mathbf{0}$

The first property allows us to formulate the derivative of the predicted state as

$$\frac{\partial}{\partial t} \hat{\mathbf{x}}_t = \dot{\mathbf{g}}(\mathbf{a}, t). \quad (11)$$

Therefore, we can calculate the gradients of the predicted state with respect to the time domain in analytical, closed form and do not require computationally expensive automatic differentiation, addressing limitation (I). Additionally, the initial condition is always fulfilled, i.e.,

$$\mathcal{L}_{\text{IC}} \equiv 0. \quad (12)$$

This addresses limitation (III) due to the second property of  $\mathbf{g}$  and introducing the initial state  $\mathbf{x}_0$  in (9). The latter is comparable to a residual neural network (ResNet) architecture (He et al., 2016). The loss-balancing problem arising from (5) is then drastically simplified. If no real data is used for training and only a fast surrogate model is to be learned, no loss balancing is even necessary, as only one loss exists. The consideration of lim. (II) can be found in section 3.2.

#### 3.1 Formulating the Ansatz Function

The Ansatz function

$$\mathbf{g} = (g_1, \dots, g_j, \dots, g_m)^\top \quad (13)$$

consists of  $m$  residual predictions  $g_j$ , where  $m$  is the number of states in the state-space model and  $g_j$  the residual prediction of the  $j$ -th state variable  $\Delta \hat{x}_{t,j}$  for a given  $t$ . The residual state prediction is constructed using  $n_g$  sub-functions.

$$g_j = \sum_{i=1}^{n_g} \alpha_{ij} (\phi_g(\beta_{ij}t + \gamma_{ij}) - \phi_g(\gamma_{ij})) \quad (14)$$

where  $\phi_g$  is a differentiable activation or base construction function. We observe that the subtrahend in (14) ensures that  $g_j(t=0) \equiv 0$ , and the total length of the vector

$$\mathbf{a} = (\boldsymbol{\alpha}^\top, \boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top)^\top \quad (15)$$

with coefficient vectors  $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$  results to  $3 \cdot m \cdot n_g$ . Finally, we obtain the derivative of the predicted state

$$\dot{g}_j = \sum_{i=1}^{n_g} \alpha_{ij} \beta_{ij} \dot{\phi}_g(\beta_{ij}t + \gamma_{ij}) \quad (16)$$

using the chain rule. Alternatively, a damping term can be added to increase the numbers of parameters in the sub-functions

$$\begin{aligned} g_{dj} &= \sum_{i=1}^{n_g} \alpha_{ij} (e^{-\delta_{ij}t} \phi_g(\beta_{ij}t + \gamma_{ij}) - \phi_g(\gamma_{ij})), \\ \dot{g}_{dj} &= \sum_{i=1}^{n_g} \alpha_{ij} e^{-\delta_{ij}t} (\beta_{ij} \dot{\phi}_g(\beta_{ij}t + \gamma_{ij}) \\ &\quad - \delta_{ij} \phi_g(\beta_{ij}t + \gamma_{ij})), \end{aligned} \quad (17)$$

where now the length of vector  $\mathbf{a} = (\boldsymbol{\alpha}^\top, \boldsymbol{\beta}^\top, \boldsymbol{\gamma}^\top, \boldsymbol{\delta}^\top)^\top$  results to  $4 \cdot m \cdot n_g$ .

The inclusion of the alternative damping term and the choice of the base construction function  $\phi_g$  may be decided based on the given state-space model characteristics or empirical testing, which is similar to the choice of a suitable activation functions of neural networks. In this study,

$$\phi_g(x) = \sin(x) \quad \text{and} \quad \dot{\phi}_g(x) = \cos(x) \quad (18)$$

is used, which achieves good results on all three different systems in Sec. 4. However, any other differentiable function may be used.

### 3.2 Higher-Order Excitation

In order to address limitation (II) concerning the zero-order hold assumption, we introduce the option for higher-order excitation input to the DD-PINN. As visualized in Fig. 1(c), the input to the FNN is the extended excitation vector  $\mathbf{u}_0^*$ , that contains a concatenation of multiple excitation vectors inside the interval  $[0, T]$ . The function  $f_u$  calculates the interpolated excitation vector

$$\mathbf{u}_t = f_u(\mathbf{u}_0^*, t) \quad (19)$$

fed into the dynamics equation  $f_{SSM}$ . In this study, we limit the analysis to zero-, first- and second-order excitation, hereinafter referred to as the degree  $\delta_u$  of  $\mathbf{u}$ , using the following polynomial interpolations with  $t_* = \frac{t}{T}$ :

1. Zero-order hold,  $\delta_u = 0$ , similar to PINC:

$$\mathbf{u}_0^* = \mathbf{u}_0, \quad f_u = \mathbf{u}_0 \quad (20)$$

2. First order,  $\delta_u = 1$ :

$$\mathbf{u}_0^* = (\mathbf{u}_0^\top, \mathbf{u}_T^\top)^\top, \quad f_u = \mathbf{u}_T t_* + (1 - t_*) \mathbf{u}_0 \quad (21)$$

3. Second order,  $\delta_u = 2$ :

$$\begin{aligned} \mathbf{u}_0^* &= (\mathbf{u}_0^\top, \mathbf{u}_{T/2}^\top, \mathbf{u}_T^\top)^\top, \\ f_u &= (2\mathbf{u}_0 - 4\mathbf{u}_{T/2} + 2\mathbf{u}_T) t_*^2 \\ &\quad + (-3\mathbf{u}_0 + 4\mathbf{u}_{T/2} - 1\mathbf{u}_T) t_* + \mathbf{u}_0. \end{aligned} \quad (22)$$

Here,  $\mathbf{u}_{T/2} = \mathbf{u}(t = \frac{T}{2})$  and  $\mathbf{u}_T = \mathbf{u}(t = T)$  denote the excitation at half and full step interval. The FNN then implicitly learns the polynomial interpolation of the components in  $\mathbf{u}_0^*$ . It can be noted that instead of this polynomial interpolation, a polynomial Taylor expansion may also be used. If applied for model predictive control, a DD-PINN trained for first-order or second-order input can still predict the system behavior under zero-order-hold assumption with  $\mathbf{u}_0 = \mathbf{u}_{T/2} = \mathbf{u}_T$ . Depending on the system's characteristics, an optimized excitation vector for the (large) time steps within MPC could be transformed into a first/second-order excitation to better approximate fine changes within small time steps outside the MPC loop.

### 3.3 Application of the DD-PINN to Control Scenarios and Its Generality

When the PINC is operated in self-loop mode for prediction of the dynamical system behavior as visualized in Fig. 1(b), only a relatively short time horizon is predicted at once compared to applications of the classical PINN. We therefore argue it is sufficiently accurate to model this short interval with the Ansatz function  $g$  of the DD-PINN without the need of a high number of construction functions  $n_g$ . The DD-PINN also maintains the generality of the PINC, as the parameter  $n_g$  as well as the FNN size can be chosen sufficiently large to model any arbitrary, continuous function. It shall also be noted, that the concept proposed here is not limited to the time domain but may be applied to the spatial domain or others for the solution of ODEs/PDEs.

## 4 VALIDATION

In this section, the PINC and the DD-PINN are evaluated on three dynamical systems visualized in Fig. 2 for comparison. The PINC and DD-PINN are both trained and implemented using PyTorch. LRA is used to balance the initial-condition loss (only PINC), the data loss, and the physics loss. A data loss is only included for the nonlinear mass-spring-damper system. It is noted that these data points are not necessary for training but demonstrate the capability of the PINC and DD-PINN to include data that may be obtained from a real system. For all systems, the Adam optimization algorithm is used with an initial learning rate of  $\alpha_{\text{init}} = 0.001$ . A learning-rate scheduler that reduces the learning rate on a plateau is employed and the training is stopped early if the rate falls below

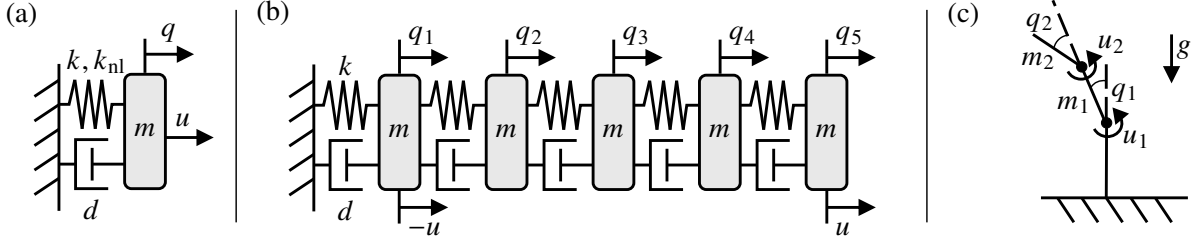


Figure 2: Three dynamical systems for evaluation: (a) nonlinear mass-spring-damper system, (b) five-mass-chain system, (c) two-link manipulator system used in (Nicodemus et al., 2022).

a certain threshold of  $\alpha_{\min} = 5 \cdot 10^{-8}$ . For the FNN in all models, the Gaussian error linear unit (GELU) activation function is used (Hendrycks and Gimpel, 2016). The first system is trained on an Intel Cascade Lake Xeon Gold 6230N CPU with 16 GB of RAM and the latter two on an Intel Core i9-10900X CPU with 64 GB of RAM. The training and neural network parameters for each system are listed in Table 1 in the appendix and an overview of the results are given in Table 2–3.

#### 4.1 Nonlinear Mass-Spring-Damper System

The nonlinear mass-spring-damper system visualized in Fig. 2(a) consists of a mass  $m$  that is connected to a fixed wall through a spring with linear stiffness  $k$  and a cubic component of  $k_{nl}$  as well as a damper with coefficient  $d$ . An external force  $u$  is applied to the mass and the displacement  $q$  and velocity  $\dot{q}$  denote the system's state  $x = (q, \dot{q})^T$ . The state-space model results to

$$\dot{x} = \begin{pmatrix} \dot{q} \\ \frac{u - d\dot{q} - kq - k_{nl}q^3}{m} \end{pmatrix} \quad (23)$$

with  $m = 0.001$  kg,  $d = 0.001$  Ns/m,  $k = 1$  N/m, and  $k_{nl} = 15$  N/m<sup>3</sup>.

Both architectures, the PINC and the DD-PINN, are evaluated for three operation frequencies  $f \in \{50\text{Hz}, 100\text{Hz}, 200\text{Hz}\}$ . The DD-PINN is further evaluated in all combinations for zero-, first-, and second-order excitation  $\delta_u \in \{0, 1, 2\}$  as described in section 3.2 and for different numbers of base construction functions  $n_g \in \{5, 20, 50\}$  – resulting in a total of 27 DD-PINN models trained. For those models, a trigonometric base function  $\phi_g(x) = \sin(x)$  is used including the optional damping factor described in (17).

For both architectures, an FNN with 96 neurons and one hidden layer is chosen. The PINC is trained for a maximum of 4,000 epochs on 20,000 collocation points, 40,000 initial condition points and 2,000 data points distributed over 400 batches. The DD-PINN is trained for a maximum of 4,000 epochs as

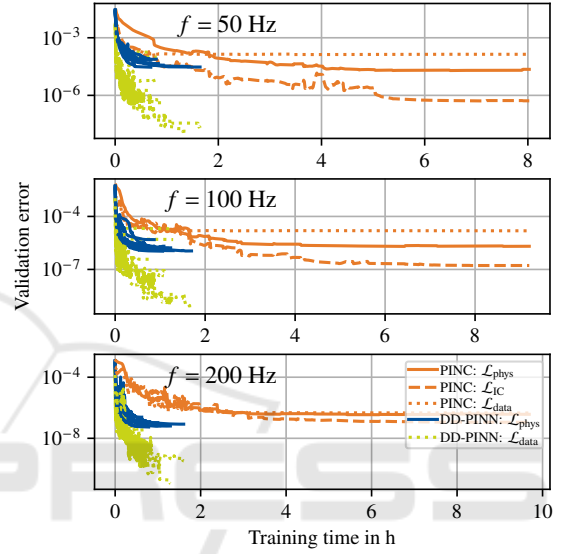


Figure 3: Nonlinear mass-spring-damper system learning; Reduction of validation errors over time. The DD-PINN training time is significantly lower.

well. The fast calculation of closed-form gradients allows us to increase the count of collocation points to 375,000 distributed over 1,500 batches. Data points are obtained from a simulation of the state-space model (23).

The sampling ranges for the collocation points

$$\begin{aligned} & (q_{\min}, \dot{q}_{\min}, u_{\min}, t_{\min}) \\ & = (-0.4 \text{ m}, -18 \text{ m/s}, -1 \text{ N}, 0 \text{ s}), \\ & (q_{\max}, \dot{q}_{\max}, u_{\max}, t_{\max}) \\ & = (0.4 \text{ m}, 18 \text{ m/s}, 1 \text{ N}, \frac{1.1}{f}) \end{aligned} \quad (24)$$

are also used for normalization of the network inputs.

The training process is visualized in Fig. 3 showing the validation physics loss  $\mathcal{L}_{\text{phys}}$  decrease over time. We observe that the DD-PINN models converge in average 8.7 times (minimum 4.7, maximum 18.9) faster than the PINC models despite a 9.375 times higher count of collocation points per epoch. The minimum validation error reached by the DD-PINN

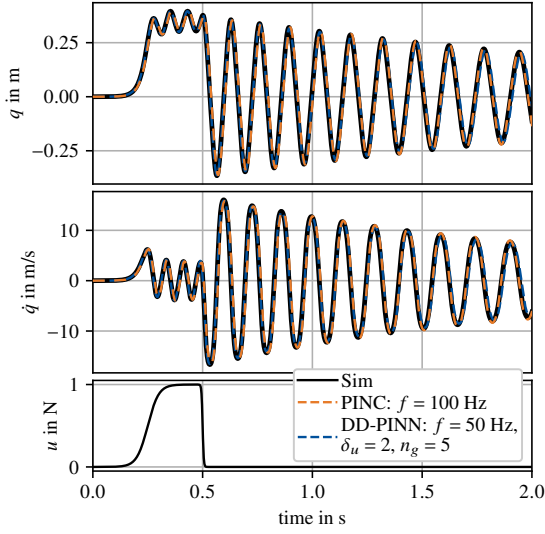


Figure 4: Nonlinear mass-spring-damper system self-loop prediction of a test path for selected PINN and DD-PINN models. Both models exhibit high prediction accuracy.

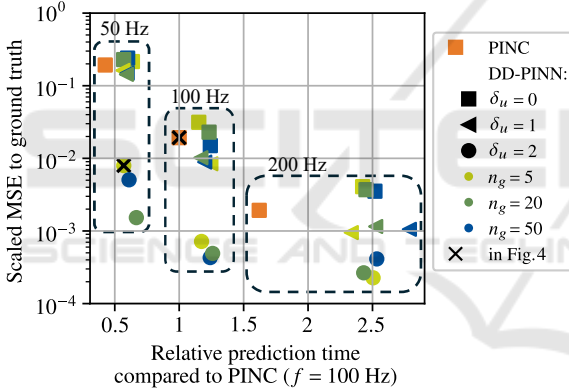


Figure 5: Parameter study of the nonlinear mass-spring-damper system learning results; Scaled MSE to ground truth of 1 s self-loop prediction path over relative prediction time. DD-PINN models achieve significantly higher prediction accuracy for slightly slower prediction time.

networks is also lower in the case of 100 and 200 Hz, but slightly higher for 50 Hz. It can also be seen that the DD-PINN reaches a drastically lower validation error for the data loss. The trained systems are evaluated on a test path in self-loop prediction of which the PINN model at  $f = 100$  Hz and a selected DD-PINN model ( $f = 50$  Hz,  $\delta_u = 2$ ,  $n_g = 5$ ) are visualized in Fig. 4. We observe an accurate and stable self-loop prediction of both models on a time horizon of more than 5 s.

A comparison of accuracy over self-loop prediction time is given in Fig. 5. Here, the MSE between the predicted path from Fig. 4 and the ground truth is calculated until  $t = 1$  s and visualized over the pre-

diction time in relation to the PINN model at  $f = 100$  Hz. We observe that a higher frequency, i.e., a shorter shooting interval generally corresponds to higher prediction accuracy, coming at cost of prediction time. Looking at each group of frequencies 50, 100, and 200 Hz respectively, the DD-PINN achieves up to 2.1, 1.7, and 0.9 magnitudes higher accuracy with  $\delta_u > 0$ . The prediction time is only slightly higher at approx. 1.2 times for the 100 Hz models and approx. 1.5 times for the other two groups. A more detailed overview of the results is also given in the appendix in Table 2.

## 4.2 Five-Mass-Chain System

The second system is a one-dimensional coupled chain of five masses as visualized in Fig. 2(b). They are connected to one side to a fixed wall and between each other with a total of five stiffnesses  $k$  and dampers  $d$ . As an external input, the same load  $u$  is applied antagonistically at the first and last mass. The state vector is defined in relative coordinates with displacements and velocities

$$\begin{aligned} \mathbf{q} &= (q_1, q_2, q_3, q_4, q_5)^\top, \\ \dot{\mathbf{q}} &= (\dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5)^\top, \end{aligned} \quad (25)$$

resulting in a total of ten states in the state vector

$$\mathbf{x} = (\mathbf{q}^\top, \dot{\mathbf{q}}^\top)^\top. \quad (26)$$

The state vector is evaluated on a physics loss function based on the equation of motion

$$\mathbf{M}\ddot{\mathbf{q}}(t) + \mathbf{D}\dot{\mathbf{q}}(t) + \mathbf{K}\mathbf{q}(t) = \mathbf{P}u(t) \quad (27)$$

with mass matrix  $\mathbf{M}$ , stiffness matrix  $\mathbf{K}$ , damping Matrix  $\mathbf{D}$ , and vector

$$\mathbf{P} = (-1, 0, 0, 0, 1)^\top. \quad (28)$$

Two DD-PINN models are trained at  $f = 50$  Hz with  $\delta_u \in \{0, 2\}$ . The hyper parameters are set to  $n_g = 20$ ,  $\phi_g(x) = \sin(x)$  and the optional damping factor given in (17) is used. This system is hard to predict accurately under the ZOH assumption showcased by one PINN model trained at  $f = 50$  Hz. For the DD-PINN and PINN, an FNN with 128 neurons and two hidden layers is chosen. The PINN is trained on 20,000 collocation points, 40,000 initial condition points over 400 batches and without data loss. The DD-PINN is trained for 2,500 epochs, using 250,000 collocation points distributed over 500 batches. The sampling ranges for the collocation points and input normalization are determined based on the simulated test path used, including a margin of 10%.

Fig. 6 shows the validation physics loss reduction during the training process. The DD-PINNs both converged at around 4 h while the PINN training was

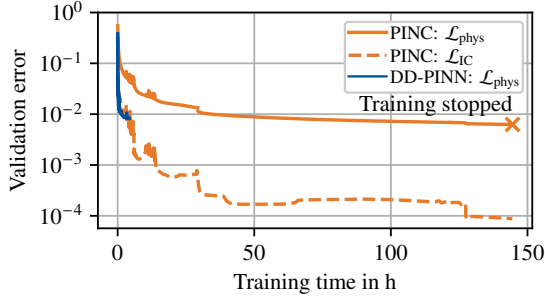


Figure 6: Five-mass-chain system learning: Reduction of validation errors over training time. DD-PINN converged within 5 h, while the PINC training was stopped after 145 h.

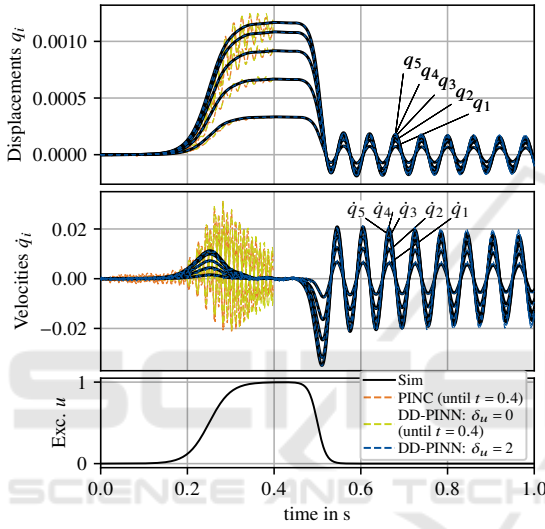


Figure 7: DD-PINN self-loop prediction of a test path for the five-mass-chain system including poor predictions for models with ZOH assumption until  $t = 0.4$ s. The DD-PINN's higher order excitation enables smooth prediction.

stopped after 145 h and 2,400 epochs of training. As we can observe in Fig. 7, the ZOH assumption is not sufficient for a prediction frequency at 50 Hz as strong oscillations are visible in both, the PINC's and the DD-PINN's self-loop prediction (only plotted until  $t = 0.4$ s). This is due to the interval-wise constant input acting as a step excitation that excites higher modes of the system.

It can be seen that having an interval-wise quadratic excitation input to the DD-PINN, the predicted path becomes stable and accurate to the simulated system response. The self-loop prediction time for the DD-PINN models was measured to be 33% and 34% slower than the PINC for  $\delta_u = 0$  and  $\delta_u = 2$ , respectively, noting that the increased order in excitation does not come along with an increase in prediction time for the DD-PINN.

### 4.3 Two-Link Manipulator System

The two-link manipulator system visualized in Fig. 2(c) denotes a Schunk PowerCube robot with dynamics identified in (Fehr et al., 2022). In previous work (Nicodemus et al., 2022), a PINC was used in MPC for fast and accurate prediction of the system dynamics. The manipulator consists of one fixed link in upright position following two sequential links connected by two joints at angle  $q_1$  and  $q_2$  at which the motor currents  $u_1, u_2$  apply. With the generalized coordinates  $\mathbf{q} = (q_1, q_2)^T$  and the state vector  $\mathbf{x} = (\mathbf{q}^T, \dot{\mathbf{q}}^T)^T$ , the state-space model results to

$$\dot{\mathbf{x}} = \begin{pmatrix} \dot{\mathbf{q}} \\ \mathbf{M}(\mathbf{q})^{-1}(\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{k}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{B}\mathbf{u}) \end{pmatrix} \quad (29)$$

with mass matrix  $\mathbf{M}$ , input matrix  $\mathbf{B}$  (including motor constant), vector of centrifugal and Coriolis forces  $\mathbf{k}$  and vector of gravitational and damping forces  $\mathbf{h}$  adopted from (Nicodemus et al., 2022).

One PINC and one DD-PINN model at  $f = 5$  Hz are trained. The DD-PINN is set to  $\delta_u = 0$  for direct comparison, using  $n_g = 20$ ,  $\phi_g(x) = \sin(x)$  and the optional damping factor given in (17). For the PINC, an FNN with 64 neurons and four hidden layers (equivalent to (Nicodemus et al., 2022)) is chosen and 128 neurons with two hidden layers for the DD-PINN. The PINC is trained for 7,000 epochs on 20,000 collocation points, 40,000 initial condition points over 400 batches and without data loss. The DD-PINN is trained for 2,500 epochs, using an increased count of collocation points of  $1 \cdot 10^6$  distributed over 1,000 batches. The used sampling ranges for the collocation points and input normalization are

$$\begin{aligned} & (q_{1,\min}, q_{2,\min}, \dot{q}_{1,\min}, \dot{q}_{2,\min}, u_{1,\min}, u_{2,\min}, t_{\min}) \\ & = (-\pi, -\pi, -1 \text{ s}^{-1}, -1 \text{ s}^{-1}, -0.6 \text{ A}, -0.6 \text{ A}, 0 \text{ s}), \\ & (q_{1,\max}, q_{2,\max}, \dot{q}_{1,\max}, \dot{q}_{2,\max}, u_{1,\max}, u_{2,\max}, t_{\max}) \\ & = (\pi, \pi, 1 \text{ s}^{-1}, 1 \text{ s}^{-1}, 0.6 \text{ A}, 0.6 \text{ A}, \frac{1.1}{f}). \end{aligned} \quad (30)$$

Fig. 9 shows the validation physics loss reduction during the training process. The training of the PINC did not converge within 7000 epochs and was discontinued due to the exceedingly high training time of over 210 h. The DD-PINN converged in a comparatively short time of 7.4 h while reaching a significantly lower validation error. This difference shows effect in the self-loop prediction on a custom test path in Fig. 8. Here, the PINC's prediction diverges after 7 s. This is comparable to previous work where the PINC prediction diverged in self-loop after approx. 3 s for the same system (Nicodemus et al., 2022). In contrast, the prediction of the DD-PINN remains stable and accurate for the whole 28 s-long test path. For



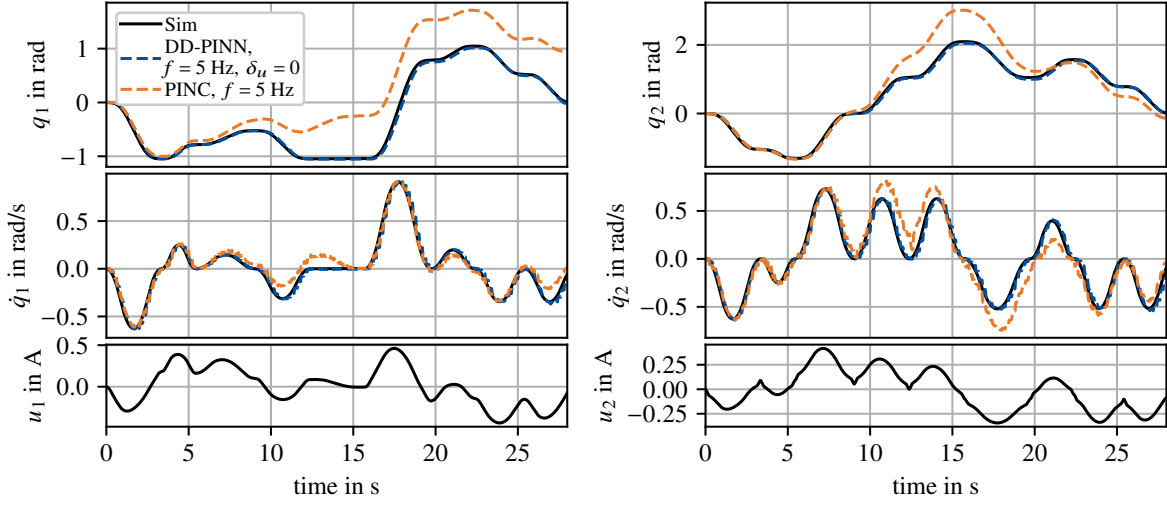


Figure 8: PINC and DD-PINN self-loop prediction of a test path for the two-link manipulator system. The DD-PINN’s prediction is stable while the PINC’s prediction diverges.

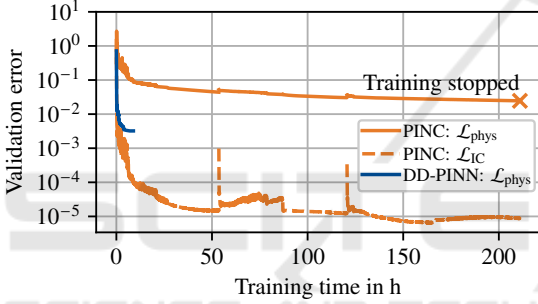


Figure 9: Two-link manipulator system learning: Reduction of validation errors over training time. DD-PINN converged within 7.4 h, while the PINC training was stopped after more than 210 h.

this system, the self-loop prediction time for the DD-PINN model was measured to be approximately equal to the computation time of the PINC. The results of the two-link manipulator system as well as the five-mass-chain system are also given in the appendix in Table 3.

#### 4.4 Discussion

The results show that the DD-PINN successfully addresses the three limitations of the PINC as formulated in Sec. 1: (I) Without the need of automatic differentiation to calculate the physics loss, *training times were reduced significantly for all three tested systems by a factor of 5–38* while evaluating 10–25 times more collocation points per epoch. A higher count of collocation points improves the coverage over the sampling ranges. In the majority of cases, the DD-PINN also reached lower physics validation

errors with a significant improvement for the two-link manipulator system that enabled *stable self-loop prediction in contrast to the PINC*. (II) The limitations of the zero-order hold assumption are overcome by introducing the option for higher-order excitation. This was shown to be necessary for the five-mass-chain system to prevent induced vibrations from the interval-wise step excitation for  $\delta_u = 0$ . *Also, a higher-order excitation in combination with a longer shooting interval enables faster prediction while being more accurate* as shown for the nonlinear mass-spring damper system. Lastly (III), the initial-condition loss for the DD-PINN is negligible ( $\mathcal{L}_{IC} \equiv 0$ ). While a loss-balancing scheme such as LRA may still be employed when the data loss is used, the *balancing process is strongly simplified*. This is because the reduction of the physics loss towards a solution of the initial-value problem for the PINC depends on a sufficiently low initial-condition loss.

The DD-PINN therefore provides a solution to these limitations while being easy to implement and interchangeable to the classical PINC architecture. Also, it is compatible with a wide range of arbitrary Ansatz functions that may be chosen suitable for the problem at hand.

## 5 CONCLUSIONS

Physics-informed neural networks (PINNs) represent a well-established machine-learning framework that combines learning the solution of differential equations with supervised learning based on data. In its adaptation to dynamical systems, namely the PINN

for control (PINC), we identified three primary limitations: (I) Long training times for large and/or complex state-space models, (II) limitations arising from the zero-order-hold assumption for excitation, and (III) the need for hyperparameter-sensitive loss balancing schemes.

In this study, we introduced the domain-decoupled physics-informed neural network (DD-PINN) as a solution to these limitations. We first formulated the DD-PINN architecture, showcasing how it enables calculation of gradients for the physics loss in closed form, is compatible to higher-order excitation input, and always has an initial-condition loss of zero. We then compared the DD-PINN to the PINC in simulation for three benchmark systems. The results demonstrated that the DD-PINN significantly reduces training times while maintaining or surpassing the prediction accuracy of the PINC. Thereby, the self-loop prediction time of the DD-PINN is comparable to the PINC.

The DD-PINN allows for fast and accurate learning of large and complex dynamical systems, which were previously out of reach for the PINC. Its fast prediction abilities create opportunities for enabling MPC in larger dynamical systems, where traditional methods like numerical integrators are too slow, and training a PINC is not practical. Here, the data efficiency of physics-informed machine learning remains, making it possible to integrate sparse datasets with the system-governing physical equations. Future work could explore using DD-PINN for higher-dimensional nonlinear systems to realize accurate state estimation or model predictive control.

## ACKNOWLEDGEMENTS

This work was partially funded by the German Research Foundation (DFG, project numbers 405032969 and 433586601) and the Lower Saxonian Ministry of Science and Culture in the program zukunft.niedersachsen.

## REFERENCES

- Antonelo, E. A., Camponogara, E., Seman, L. O., Jordanou, J. P., de Souza, E. R., and Hübner, J. F. (2024). Physics-informed neural nets for control of dynamical systems. *Neurocomputing*, 579:127419.
- Berg, J. and Nyström, K. (2018). A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41.
- Bischof, R. and Kraus, M. (2021). Multi-objective loss balancing for physics-informed deep learning. *arXiv preprint arXiv:2110.09813*.
- Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. (2018). GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR.
- Chiu, P.-H., Wong, J. C., Ooi, C., Dao, M. H., and Ong, Y.-S. (2022). CAN-PINN: A fast physics-informed neural network based on coupled-automatic-numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909.
- Cuomo, S., Di Cola, V. S., Giampaolo, F., Rozza, G., Raissi, M., and Piccialli, F. (2022). Scientific machine learning through physics-informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88.
- de Curtò, J. and de Zarzà, I. (2024). Hybrid state estimation: Integrating physics-informed neural networks with adaptive ukf for dynamic systems. *Electronics*, 13(11):2208.
- Deguchi, S. and Asai, M. (2023). Dynamic & norm-based weights to normalize imbalance in back-propagated gradients of physics-informed neural networks. *Journal of Physics Communications*, 7(7):075005.
- Fehr, J., Kargl, A., and Eschmann, H. (2022). Identification of friction models for MPC-based control of a power-cube serial robot. *arXiv preprint arXiv:2203.10896*.
- Gupta, G., Xiao, X., and Bogdan, P. (2021). Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062.
- Hao, Z., Liu, S., Zhang, Y., Ying, C., Feng, Y., Su, H., and Zhu, J. (2022). Physics-informed machine learning: A survey on problems, methods and applications. *arXiv preprint arXiv:2211.08064*.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.
- Heydari, A. A., Thompson, C. A., and Mehmood, A. (2019). Softadapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions. *arXiv preprint arXiv:1912.12355*.
- Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *Journal of Computational Physics*, 404:109136.
- Kapoor, T., Wang, H., Núñez, A., and Dollevoet, R. (2024). Physics-informed neural networks for solving forward and inverse problems in complex beam systems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(5):5981–5995.
- Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. (2021). Physics-informed

- machine learning. *Nature Reviews Physics*, 3(6):422–440.
- Kharazmi, E., Zhang, Z., and Karniadakis, G. E. (2019). Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*.
- Kharazmi, E., Zhang, Z., and Karniadakis, G. E. M. (2021). Hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547.
- Lagaris, I., Likas, A., and Fotiadis, D. (Sept./1998). Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000.
- Liu, C. and Wu, H. (2023). Cv-PINN: Efficient learning of variational physics-informed neural network with domain decomposition. *Extreme Mechanics Letters*, 63:102051.
- Nghiem, T. X., Drgoňa, J., Jones, C., Nagy, Z., Schwan, R., Dey, B., Chakrabarty, A., Di Cairano, S., Paulson, J. A., Carron, A., et al. (2023). Physics-informed machine learning for modeling and control of dynamical systems. In *2023 American Control Conference (ACC)*, pages 3735–3750. IEEE.
- Nicodemus, J., Kneiff, J., Fehr, J., and Unger, B. (2022). Physics-informed neural networks-based model predictive control for multi-link manipulators. *IFAC-PapersOnLine*, 55(20):331–336.
- Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707.
- Sanyal, S. and Roy, K. (2023). RAMP-net: A robust adaptive MPC for quadrotors via physics-informed neural network. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1019–1025. IEEE.
- Tang, S., Feng, X., Wu, W., and Xu, H. (2023). Physics-informed neural networks combined with polynomial interpolation to solve nonlinear partial differential equations. *Computers & Mathematics with Applications*, 132:48–62.
- Wang, S., Sankaran, S., and Perdikaris, P. (2022). Respecting causality is all you need for training physics-informed neural networks. *arXiv preprint arXiv:2203.07404*.
- Wang, S., Sankaran, S., Wang, H., and Perdikaris, P. (2023). An expert’s guide to training physics-informed neural networks. *arXiv preprint arXiv:2308.08468*.
- Wang, S., Teng, Y., and Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081.
- Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. (2023). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 403:115671.
- Yang, X., Du, Y., Li, L., Zhou, Z., and Zhang, X. (2023). Physics-informed neural network for model prediction and dynamics parameter identification of collaborative robot joints. *IEEE Robotics and Automation Letters*, 8(12):8462–8469.
- Yang, Y., Hou, M., Sun, H., Zhang, T., Weng, F., and Luo, J. (2020). Neural network algorithm based on Legendre improved extreme learning machine for solving elliptic partial differential equations. *Soft Computing*, 24:1083–1096.
- Zhang, Y., Wang, M., Zhang, F., and Chen, Z. (2023). A solution method for differential equations based on Taylor pinn. *IEEE Access*, 11:145020–145030.

## APPENDIX

Table 1: PINC and DD-PINN training and neural-network parameters for the three evaluated systems.

	Nonlinear mass-spring-damper		Five-mass chain		Two-link manipulator	
	PINC	DD-PINN	PINC	DD-PINN	PINC	DD-PINN
Epochs	$\leq 4,000$	$\leq 4,000$	2,400	2,500	7,000	2,500
Batches	400	1,500	400	500	400	1,000
$n_{IC}$	40,000	0	40,000	0	40,000	0
$n_{collo}$	20,000	375,000	20,000	250,000	20,000	$1 \cdot 10^6$
$n_{data}$	2,000	2,000	0	0	0	0
Neurons	96	96	128	128	64	128
Hidden layers	1	1	2	2	4	2

Table 2: PINC and DD-PINN training results for the nonlinear mass-spring-damper. Best values are highlighted in bold.

	Nonlinear mass-spring-damper					
	PINC			DD-PINN*		
Number of models trained	1	1	1	9	9	9
$f$ in Hz	50	100	200	50	100	200
Training time in h	8.04	9.22	9.70	0.64	0.76	<b>0.54</b>
$\mathcal{L}_{IC}$	$5.20 \cdot 10^{-7}$	$1.67 \cdot 10^{-7}$	$1.11 \cdot 10^{-7}$	<b>0</b>	<b>0</b>	<b>0</b>
$\mathcal{L}_{phys}$	$2.26 \cdot 10^{-5}$	$2.05 \cdot 10^{-6}$	$3.72 \cdot 10^{-7}$	$2.90 \cdot 10^{-5}$	$1.08 \cdot 10^{-6}$	<b><math>5.49 \cdot 10^{-8}</math></b>
$\mathcal{L}_{data}$	$1.38 \cdot 10^{-4}$	$1.54 \cdot 10^{-5}$	$4.55 \cdot 10^{-7}$	$7.67 \cdot 10^{-5}$	$4.87 \cdot 10^{-6}$	$2.42 \cdot 10^{-7}$
Scaled MSE on test path	$1.93 \cdot 10^{-1}$	$1.92 \cdot 10^{-2}$	$1.92 \cdot 10^{-3}$	$1.21 \cdot 10^{-8}$	$6.79 \cdot 10^{-10}$	<b><math>1.15 \cdot 10^{-11}</math></b>
Relative pred. time	<b>0.43</b>	1.00	1.62	$1.78 \cdot 10^{-4}$	$1.99 \cdot 10^{-5}$	$7.83 \cdot 10^{-8}$
				$1.53 \cdot 10^{-3}$	$4.30 \cdot 10^{-4}$	<b><math>2.26 \cdot 10^{-4}</math></b>
				$2.40 \cdot 10^{-1}$	$3.14 \cdot 10^{-2}$	$4.07 \cdot 10^{-3}$
				0.57	1.15	2.34
				0.67	1.26	2.79

\*Values denote min and max values among the different configurations with varying degree of excitation  $\delta_u \in \{0, 1, 2\}$  and different numbers of base construction functions  $n_g \in \{5, 20, 50\}$ .

Table 3: PINC and DD-PINN training results for the five-mass-chain and two-link manipulator systems. Best values per system are highlighted in bold.

	Five-mass chain		Two-link manipulator	
	PINC	DD-PINN*	PINC	DD-PINN
Training time in h	144.5	4.29	211.0	<b>7.41</b>
$\mathcal{L}_{IC}$	$8.80 \cdot 10^{-5}$	<b>0</b>	$8.90 \cdot 10^{-6}$	<b>0</b>
$\mathcal{L}_{phys}$	<b><math>6.27 \cdot 10^{-3}</math></b>	$8.43 \cdot 10^{-3}$	$2.46 \cdot 10^{-2}$	<b><math>3.22 \cdot 10^{-3}</math></b>
$\mathcal{L}_{data}$		$8.22 \cdot 10^{-3}$		
Scaled MSE on test path	$1.78 \cdot 10^{-1}$	$2.42 \cdot 10^{-1}$	$9.17 \cdot 10^{-2}$	<b><math>1.68 \cdot 10^{-3}</math></b>
Relative pred. time	<b>1.00</b>	1.33	<b>1.00</b>	1.03
		1.34		

\*Values for the DD-PINN for the five-mass-chain system correspond to the configurations with  $\delta_u = 0$  and  $\delta_u = 2$ .