Taylor-Model Physics-Informed Neural Networks (PINNs) for Ordinary Differential Equations

Chandra Kanth Nagesh and **Sriram Sankaranarayanan**   FIRST.LASTNAME@COLORADO.EDU
University of Colorado Boulder

Ramneet Kaur, Tuhin Sahai and **Susmit Jha**   FIRST.LASTNAME@SRI.COM
SRI International

Editors: G. Pappas, P. Ravikumar, S. A. Seshia

Abstract

We study the problem of learning neural network models for Ordinary Differential Equations (ODEs) with parametric uncertainties. Such neural network models capture the solution to the ODE over a given set of parameters, initial conditions, and range of times. Physics-Informed Neural Networks (PINNs) have emerged as a promising approach for learning such models that combine data-driven deep learning with symbolic physics models in a principled manner. However, the accuracy of PINNs degrade when they are used to solve an entire family of initial value problems characterized by varying parameters and initial conditions.

In this paper, we combine symbolic differentiation and Taylor series methods to propose a class of higher-order models for capturing the solutions to ODEs. These models combine neural networks and symbolic terms: they use higher order Lie derivatives and a Taylor series expansion obtained symbolically, with the remainder term modeled as a neural network. The key insight is that the remainder term can itself be modeled as a solution to a first-order ODE. We show how the use of these higher order PINNs can improve accuracy using interesting, but challenging ODE benchmarks. We also show that the resulting model can be quite useful for situations such as controlling uncertain physical systems modeled as ODEs.

Keywords: Physics-Informed Neural Networks, Initial Value Problems, Ordinary Differential Equations, Taylor Models

1. Introduction

Finding closed-form analytic solutions to systems of Ordinary Differential Equations (ODEs) is challenging for all but the simplest class of systems. The problem is even more challenging for ODEs with parameters that can take on a set of possible values, unknown initial conditions and external inputs. Physics-Informed Neural Networks (PINNs) have emerged as a solution to finding approximate closed-forms modeled as neural networks Raissi et al. (2018). They have been studied for solving PDEs, especially non-linear PDEs that are hard to solve numerically. The problem of PINNs for ODEs have received considerably less attention since numerical solvers are quite successful in finding solutions for many common ODEs Hairer et al. (1993). However, the use of numerical solvers is distinctly problematic for applications that involve solving optimization problems involving ODEs with *unknown parameters and inputs*. These arise in machine learning, where one wishes to learn parameters from data or optimal control, wherein we seek control inputs that optimize a function across the trajectories of the system. For such applications, it is desirable to have a "surrogate model" that can capture the solution of the ODEs with high enough accuracy over a range of parameters, initial conditions and times. If each evaluation of the surrogate can be performed more efficiently than using a numerical solver, the overall optimization can be faster.

We investigate the use of PINNs to build surrogate models that capture the solution to an initial value problem (IVP) given by a system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{\theta}, t)$ for parameters $\boldsymbol{\theta} \in \Theta$, initial conditions $\boldsymbol{x}(0) \in \Omega$ and $t \in [0, T]$. In other words, our surrogate model $\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ maps the inputs to a solution $\boldsymbol{x}(t; \boldsymbol{x}_0, \boldsymbol{\theta})$ of the ODE at time t. The standard PINN approach of Raissi et al. (2019) a) uses a neural network to represent φ and b) a combination of two loss functions given by the initial condition loss $\|\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) - \boldsymbol{x}_0\|$ and the gradient loss $\|\dot{\varphi} - f(\varphi, \boldsymbol{\theta}, t)\|$ averaged at various randomly chosen "collocation points".

In this paper, we first point out the inadequacy of the PINN approach to this problem by demonstrating its failure to approximate the solution when the sets Θ, Ω and $[0, T]$ are large. We show that higher-order loss functions fail to address the issue. Therefore, we resort to a symbolic approach that uses successive Lie derivatives to compute the terms of a Taylor series expansion of the solution. We show that the remainder when carefully modeled can be written down as the solution to a derived ODE. Solving this derived ODE for the higher-order remainder using the "classic" PINN approach yields a solution that combines the best aspects of symbolic differentiation with neural network learning. We show that our error grows as $O(t^{m+1}e^{Kt})$ for an approach that uses derivatives up to order $m \geq 1$, whereas, for PINNs, the error grows as $O(te^{Kt})$. As a result, our approach provides high levels of accuracy at the initial times. We compare our approach, which we call "Taylor-Model PINNs" with PINNs, and the related approach of "Higher-Order PINNs" that extends the original PINN loss function with higher order derivative-based loss functions. We show that Taylor-Model PINNs provide higher accuracy. While our approach increases the complexity of the training process, the use of efficient symbolic differentiation tools offsets this process.

1.1. Related Work

Machine learning approaches have found applications in diverse domains ranging from celestial object classification Angeloudi et al. (2024), climate forecasting Iglesias-Suarez et al. (2024), and tumor identification Li et al. (2023). In such scenarios, it has been observed that using background knowledge in the form of mathematical models in the learning process can considerably speed up convergence and improve solution quality.

Physics-Informed Neural Networks (PINNs) Raissi et al. (2019) represent a seminal contribution in this space. They have been effective in solving systems which involve partial differential equations (PDEs), where data is sparsely available. By utilizing differentiable loss functions, a neural network is trained on the PDE residual and boundary condition loss to learn a solution map to the PDE. These physics-inspired loss terms act as a regularizer against learning solution maps that do not involve the underlying dynamics of the system, thereby conforming well to how the system evolves over time. This promising approach has led to widespread use of the methodology in various applications Shukla et al. (2020), Wang and Perdikaris (2021), Yin et al. (2021).

Despite their contributions, PINN methodologies can often fail to learn physical dynamics in many cases Krishnapriyan et al. (2021), Steger et al. (2022). To tackle such issues, there have been efforts into PINNs with additional loss functions Son et al. (2023), Wang et al. (2022). However, their efficacy diminishes under two conditions (a) conflicting gradient updates between the two loss functions, leading to suboptimal gradient descent Hwang and Lim (2025) and (b) when applied to parametric PDE families, particularly those requiring simultaneous resolution across a range of initial conditions and parameters Xiang et al. (2025).

The Taylor series expansion represents a fundamental concept in mathematical analysis, providing a powerful framework for approximating functions through polynomials or power-series Apostol (1991). Our approach of using Taylor series expansions has been heavily influenced by the work of Makino and Berz, who have applied so-called "Taylor-model calculus" to represent a set of complex and unknown functions by a finite Taylor series expansion with an interval remainder Berz and Makino (1998); Makino and Berz (2009). This has led to popular approaches in the area of formal methods for proving properties of Cyber-Physical Systems Chen and Sankaranarayanan (2022); Althoff et al. (2021); Althoff (2015); Kong et al. (2015). Here, we adapt Taylor models to represent solutions but let the remainder be represented by a neural network rather than an interval. Parts of this problem have been investigated before, where trained neural networks are approximated using Taylor polynomials to enable integration of physical constraints into dynamical systems Zhu et al. (2022), Balduzzi et al. (2016). Furthermore, researchers have looked into the idea of Taylor layers for Transformer architectures Zwerschke et al. (2024), which are higher order polynomial approximation replacements of standard linear or attention layers. The contributions of this paper can be summarized as follows:

1. We show how a symbolic Taylor series expansion of an *a priori* fixed order and the remainder term modeled by a neural network can be used to capture solutions of ODEs accurately.
2. We present an evaluation of our approach based on how PINN errors grows over time.
3. We develop a novel neural network training method and show that this new approach converges to tight solution maps compared to traditional PINNs, on seven ODE systems with varying dimensionality and parameter space.
4. The trends predicted by our analysis are empirically demonstrated on a set of examples.

2. Preliminaries

We will present some preliminary facts about ODEs, their solutions, and Taylor series expansions.

Definition 1 (Ordinary Differential Equations) *A system of (coupled) Ordinary Differential Equations (ODE) over state variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ and parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ is of the form $\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, \boldsymbol{\theta}, t)$, wherein t represents the time variable, and $f : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ is a vector-field that maps states, parameters and time to the value of the derivative.*

For an initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and fixed values of the parameters $\boldsymbol{\theta}$, the solution of the ODE is a differentiable map $\psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ such that for all time t, $\frac{d\psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)}{dt} = f(\psi(x_0, \boldsymbol{\theta}, t), \boldsymbol{\theta}, t)$. We assume that the function f defining the RHS of the ODE is *Lipschitz* continuous, thus ensuring the existence and uniqueness of solutions.

Example 1 (Duffing Oscillator) *Consider an ODE that models the dynamics of a Duffing oscillator with $\boldsymbol{x} = (x, y)$, $\boldsymbol{\theta} = (\delta)$ and dynamics given by $\frac{dx}{dt} = y$, $\frac{dy}{dt} = x - x^3 - \delta y$.*

Given an ODE model, we seek to represent the solution φ as a function of all initial conditions $\boldsymbol{x}_0$, parameters θ and time t. However, this sort of *analytical solution* is only available for a restricted class of ODEs. In practice, we have to settle for an *approximate solution* available either through a numerical ODE solver (for instance, using the Runge-Kutta algorithm) or an approximate analytic solution $\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ that provides a solution close to the real solution for a range of initial conditions $\boldsymbol{x}_0 \in \Omega, \boldsymbol{\theta} \in \Theta$ and $t \in [0, T]$ for given sets Ω, Θ and time horizon $T > 0$. The approximate solution

map has two potential advantages: (a) it can be computationally less expensive than numerical solvers; and (b) it can be used to estimate derivatives such as $\frac{d\varphi(t)}{d\boldsymbol{\theta}}$ and $\frac{d\varphi(t)}{d\boldsymbol{x}_0}$ efficiently. Computing such derivatives is useful for learning parameters from data, and is hard to do using numerical solvers. The main problem statement for this paper is as follows:

Definition 2 (Learning Solution Map) *Given the description of an ODE $\frac{d\boldsymbol{x}}{dt} = f(\boldsymbol{x}, \boldsymbol{\theta}, t)$, a set of initial conditions $\boldsymbol{x}_0 \in \Omega$, a set of parameters $\boldsymbol{\theta} \in \Theta$ and a time horizon $t \in [0, T]$ for $T > 0$, we wish to learn a solution map $\varphi : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$ such that $\varphi(\boldsymbol{x}_0, \theta, t)$ is as close as possible to the precise solution map ψ.*

The physics-informed approach learns φ by fixing a finite sample set of "collocation points" $S \subseteq \Omega \times \Theta \times [0, T]$, wherein $|S| = N$, and minimizing two different loss functions simultaneously:

$$L_i = \frac{1}{N} \sum_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in S} \|\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) - \boldsymbol{x}_0\|,$$
$$L_g = \frac{1}{N} \sum_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in S} \|\frac{\partial}{\partial t}\varphi(\boldsymbol{x}_0, \varphi, t) - f(\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, t), \boldsymbol{\theta}, t)\|$$

In practice, the approaches minimizes a linear combination $L = \omega_i L_i + \omega_g L_g$ for user-specified constants ω_i, ω_g. The process of learning φ proceeds by fixing a neural network architecture with unknown weights W and using stochastic gradient descent (since N is typically very large) technique to find a local minimizer W^* for the overall loss L.
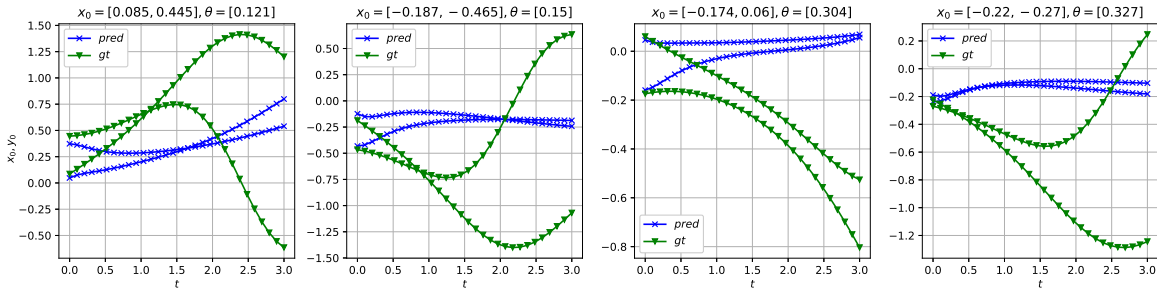

Figure 1: Numerical simulations (taken as ground-truth) shown in green compared against the PINN prediction shown in blue for the Duffing Oscillator system provided in Ex. 1. The initial conditions and parameters are randomly sampled from $\Omega \in [-0.5, 0.5]$ and $\theta \in [0.1, 0.5]$ respectively.

Example 2 *Consider the Duffing oscillator model from Ex. 1. Fig. 1 compares the "ground truth" trajectories obtained through a numerical simulation against the predictions of the PINN model given by a neural network with 1 layers and 64 neurons per layer. Note that the trajectories diverge rapidly from predictions. Also, note that since the loss L_i is not zero, it causes discrepancies even in the initial predictions.*

Let us assume that we are able to learn a neural network model $\varphi_N(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ for $\boldsymbol{x}_0 \in \Omega, \boldsymbol{\theta} \in \Theta$ and $t \in [0, T]$ such that φ_N is a differentiable function of t, and the following inequalities hold:

$$\max_{\boldsymbol{x}_0 \in \Omega, \boldsymbol{\theta} \in \Theta} \|\varphi_N(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) - \boldsymbol{x}_0\| \leq L_{i,\max} \text{ and } \max_{\boldsymbol{x}_0 \in \Omega, \boldsymbol{\theta} \in \Theta, t \in [0,T]} \|\dot{\varphi}_N(\boldsymbol{x}_0, \boldsymbol{\theta}, t) - f(\varphi_N, \boldsymbol{\theta}, t)\| \leq L_{g,\max}$$

for some constants $L_{i,\max}, L_{g,\max} > 0$. Let $\psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ represent the analytical solution of the ODE.

4

Theorem 3 *There exists a constant $K > 0$ such that, for all $\boldsymbol{x}_0 \in \Omega, \boldsymbol{\theta} \in \Theta$ and $t \in [0, T]$, we have $\|\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, t) - \psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)\| \leq (L_{i,\max} + L_{g,\max}t)e^{Kt}$*

Proof Note that $\psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t) = \boldsymbol{x}_0 + \int_0^t f(\psi(\boldsymbol{x}_0, \boldsymbol{\theta}, s), \boldsymbol{\theta}, s)ds$. Likewise, assuming differentiability of φ_N, we have $\varphi_N(\boldsymbol{x}_0, \boldsymbol{\theta}, t) = \varphi_N(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) + \int_0^t \dot{\varphi}_N(\boldsymbol{x}_0, \boldsymbol{\theta}, s)ds$. For simplicity, we write $\varphi_N(t) := \varphi_N(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ and $\psi(t) := \psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$. We have,

$$\|\varphi_N(t) - \psi(t)\| \leq \|\varphi_N(0) - \boldsymbol{x}_0\| + \left\|\int_0^t \dot{\varphi}_N(s) - f(\psi(s), \boldsymbol{\theta}, s)ds\right\|$$

$$\leq K_{i,\max} + \int_0^t \|\dot{\varphi}_N(\boldsymbol{x}_0, \boldsymbol{\theta}, s) - f(\psi, \boldsymbol{\theta}, s)\|ds$$

$$\leq K_{i,\max} + \int_0^t \|\dot{\varphi}_N(s) - f(\varphi_N(s), \boldsymbol{\theta}, s)\|ds + \int_0^t \|f(\varphi_N, \boldsymbol{\theta}, s) - f(\psi, \boldsymbol{\theta}, s)\|ds$$

$$\leq K_{i,\max} + K_{g,\max}t + L_f \int_0^t \|\varphi_N(s) - \psi(s)\|ds$$

wherein L_f is the Lipschitz constant of $f(\boldsymbol{x}, \boldsymbol{\theta}, t)$ over, $\boldsymbol{x} \in X$ obtained as $\varphi(\Omega, \Theta, [0, T]) \cup \psi(\Omega, \Theta, [0, T])$. This set will be compact if Ω, Θ are compact and T is finite. Applying Grönwall's inequality Bellman (1943), we conclude, $\|\varphi_N - \psi\| \leq (K_{i,\max} + K_{g,\max}t)e^{L_f t}$. ∎

3. Higher-Order PINNs

In this section, we will tackle the problem of using symbolic differentiation computations based on Taylor series methods. Given ODE $\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{\theta}, t)$, recall that the Lie derivative of a function $g(\boldsymbol{x}, t)$ is given by $\mathcal{L}_f(g) = (\nabla_{\boldsymbol{x}} g) \cdot f + \frac{\partial}{\partial t} g$. We define the successive Lie derivatives: $\mathcal{L}^{(0)}(\boldsymbol{x}) = f_0(\boldsymbol{x}) = \boldsymbol{x}$, and $\mathcal{L}^{(i+1)}(\boldsymbol{x}) = \mathcal{L}(f_i(\boldsymbol{x}, \boldsymbol{\theta}, t)) := f_{i+1}(\boldsymbol{x}, \boldsymbol{\theta}, t)$.

The main idea behind higher order PINNs is to consider loss functions beyond the first order loss function. For instance, the second and third order losses are defined as:

$$L_2 = \frac{1}{N} \sum_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in S} \left\|\frac{d^2}{dt^2}\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, t) - f_2(\varphi, \boldsymbol{\theta}, t)\right\| \text{ and}$$
$$L_3 = \frac{1}{N} \sum_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in S} \left\|\frac{d^3}{dt^3}\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, t) - f_3(\varphi, \boldsymbol{\theta}, t)\right\|$$

The overall loss is obtained by combining the initial condition loss L_i, the PINN gradient loss L_g, and the higher order losses: $L = \alpha_0 L_i + \alpha_1 L_g + \alpha_2 L_2 + \alpha_3 L_3 + \cdots + \alpha_m L_m$.

However, such a scheme has two main disadvantages: (a) it requires us to take higher order derivatives of a large and complex neural network model; and (b) it introduces multiple loss functions, all of which need to be minimized by selecting an appropriate linear combination of loss functions. We propose, instead, a simpler scheme based on Taylor series that has the advantage of (a) using a single loss function, (b) not requiring Hessians or higher-order gradients of neural networks and (c) tries to match the flow up to some order $m > 0$.

3.1. Higher-Order PINNs based on Taylor series

We will assume through the rest of this paper that the RHS function f is at least $m + 2$ times differentiable for some $m > 0$, we have:

$$\boldsymbol{x}(t) = \psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t) = \boldsymbol{x}_0 + tf_1(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) + \frac{t^2}{2!}f_2(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) + \cdots + \frac{t^m}{m!}f_m(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) + T_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$$

wherein T_m denotes the remainder term of order $m + 1$.

Theorem 4 *The function T_m satisfies the following properties:*

1. $T_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t) = \frac{1}{m!} \int_0^t (t - s)^m f_{m+1}(\boldsymbol{x}(s), \boldsymbol{\theta}, s) ds$

2. $T_m(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) = \dot{T}_m(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) = \cdots = T_m^{(m)}(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) = 0$

Proof The proof of the first statement is available from (Apostol, 1991, Theorem 7.6). The second statement follows from repeatedly differentiating the RHS of the first equality using Leibnitz's rule for differentiation under the integral sign. ∎

Rather than use loss functions to enforce that $T_m^{(j)}(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) = 0$ for $j \leq m$, we can write

$$T_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t) = \frac{t^{m+1}}{(m+1)!} R_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t), \text{ wherein } R_m = (m+1) \int_0^t \frac{1}{t} \left(1 - \frac{s}{t}\right)^m f_{m+1}(\boldsymbol{x}(s), \boldsymbol{\theta}, s) ds$$

R_m can be re-written using the change of variables $\alpha = \frac{s}{t}$ as

$$R_m = (m+1) \int_0^1 (1 - \alpha)^m f_{m+1}(\boldsymbol{x}(\alpha t), \boldsymbol{\theta}, \alpha t) d\alpha \tag{1}$$

The goal is to use a neural network model for R_m while inferring its parameters through a loss function. We propose two approaches: (a) an indirect approach based on the PINN loss function and (b) a direct approach that uses quadrature to approximate the integral in Eq. (1).

PINN-based loss function: We will use a PINN to model $R_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$. For convenience, we will assume that $\boldsymbol{x}_0, \theta$ are fixed and denote $\tau_m(t) = T_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$, $r_m(t) = R_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ and $f_i(0)$ denote $f_i(\boldsymbol{x}_0, \boldsymbol{\theta}, 0)$. Note that $\tau_m(t) = \frac{t^{m+1}}{(m+1)!} r_m(t)$. Let

$$\varphi_r(\boldsymbol{x}_0, \boldsymbol{\theta}, t) = f_0(0) + t f_1(0) + \cdots + \frac{t^m}{m!} f_m(0) + \tau_m(t) \tag{2}$$

Theorem 5 *The remainder τ_m is a solution to the ODE with Lipschitz continuous RHS:*

$$\dot{\tau}_m(t) = f_1(\varphi_r, \boldsymbol{\theta}, t) - \left(f_1(0) + t f_2(0) + \cdots + \frac{t^{m-1}}{(m-1)!} f_m(0) \right).$$

Furthermore, it has the form $\tau_m(t) = \frac{t^{m+1}}{(m+1)!} r_m(t)$ for a continuous and differentiable function $r_m(t)$ with $r_m(0) = f_{m+1}(\boldsymbol{x}_0, \boldsymbol{\theta}, 0)$

Now, we can use PINNs to learn the remainder R_m as a function of time using the loss functions:

1. $L_{r,g} = \frac{1}{N} \sum_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in S} \left\| f(\varphi_r, \theta, t) - \left(f_1 + t f_2 + \cdots + \frac{t^m}{(m)!} R_m + \frac{t^{m+1}}{(m+1)!} \dot{R}_m \right) \right\|$, and

2. $L_{r,i} = \frac{1}{N} \sum_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in S} \| f_{m+1}(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) - R_m(\boldsymbol{x}_0, \boldsymbol{\theta}, 0) \|$

Example 3 *Consider the Duffing Oscillator case from Ex. 1 with state variables x, y and parameter δ. The overall system $\varphi(\boldsymbol{x}_0, \delta, t)$ based on Taylor series expansion of order $m = 4$ is as follows:*

$$\begin{bmatrix} x_0 + t y_0 + \frac{t^2}{2!}(x_0 - \delta y_0 - x_0^3) + \frac{t^3}{3!} f_2(\boldsymbol{x}_0, \delta, t) + \frac{t^4}{4!} R_m(\boldsymbol{x}_0, \delta, t) \\ y_0 + t(x - \delta y - x^3) + \frac{t^2}{2!} f_2(\boldsymbol{x}_0, \delta, t) + \frac{t^3}{3!} f_3(\boldsymbol{x}_0, \delta, t) + \frac{t^4}{4!} R_m(\boldsymbol{x}_0, \delta, t) \end{bmatrix}$$

where, $f_2(\boldsymbol{x}_0, \delta, t) = (-\delta(x_0 - \delta y_0 - x_0^3) + y_0(1 - 3x_0^2))$; $f_3(\boldsymbol{x}_0, \delta, t) = (y_0(-\delta(1 - 3x_0^2) - 6x_0y_0) + (\delta^2 - 3x_0^2 + 1)(-\delta y_0 - x_0^3 + x_0))$ *are the second and third derivatives of the system and* R_m *is the neural network model that is trained to learn the remainder term of the expansion.*

Now, in order for the solution of the system φ *to match the original Duffing Oscillator, we can write* $\dot{\varphi}(\boldsymbol{x}_0, \delta, t)$ *as:*

$$\begin{bmatrix} y_0 + t(x_0 - \delta y_0 - x_0^3) + \frac{t^2}{2!}f_2(\boldsymbol{x}_0, \delta, t) + \frac{t^3}{3!}R_m(\boldsymbol{x}_0, \delta, t) + \frac{t^4}{4!}\dot{R}_m(\boldsymbol{x}_0, \delta, t) \\ (x_0 - \delta y_0 - x_0^3) + tf_2(\boldsymbol{x}_0, \delta, t) + \frac{t^2}{2!}f_3(\boldsymbol{x}_0, \delta, t) + \frac{t^3}{3!}R_m(\boldsymbol{x}_0, \delta, t) + \frac{t^4}{4!}\dot{R}_m(\boldsymbol{x}_0, \delta, t) \end{bmatrix}$$

where, $\dot{R}_m$ *is the first-order time derivative of the neural network model. The overall loss function for the learning procedure,* $L = L_{r,g} + L_{r,i}$ *can now be calculated with the both sides of the equation computed as above. The implementation of the training algorithm is provided in Appendix* D.

Loss Function Through Numerical Quadrature: Rather than differentiate R_m, we can use a numerical approach to directly encode the remainder formula in Eq. (1). Let us subdivide the interval $\alpha \in [0, 1]$ into $K + 1$ quadrature points, wherein $\alpha_k = \frac{k}{K}$ for $k \in \{0, \ldots, K\}$. Using the trapezoidal rule, we obtain

$$R_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \approx \frac{m+1}{K}\left(\frac{1}{2}F(0) + \sum_{k=1}^{K-1} F\left(\frac{k}{K}\right) + \frac{1}{2}F(1)\right) \tag{3}$$

wherein $F(\alpha) := (1 - \alpha)^m f_{m+1}(\varphi(\boldsymbol{x}_0, \boldsymbol{\theta}, \alpha t), \boldsymbol{\theta}, \alpha t)$. Note that $F(1) = 0$. However, complexity of this approach depends on the choice of K: a small value of K makes the quadrature highly erroneous whereas a larger value makes the approach quite expensive. Further investigation shows us that the quadrature method does not work well for smaller choices of K, however, is faster than our other approach. Detailed results and analysis of this behaviour is provided in Appendix B.

Analysis: Let us assume that we have inferred a differentiable model R_m which achieves a maximum possible loss $\max_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in \mathcal{S}} L_{r,g} = K_{rg,\max}$ and $\max_{(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in \mathcal{S}} L_{r,i} = K_{r,i,\max}$ over the compact set of inputs $\mathcal{S} = \Omega \times \Theta \times [0, T]$.

Let $\psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)$ represent the solution map for the ODE and φ_r denote the model from Eq. (2).

Theorem 6 *There exists a constant K such that for all $(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \in \mathcal{S}$,*

$$\|\varphi_r(\boldsymbol{x}_0, \boldsymbol{\theta}, t) - \psi(\boldsymbol{x}_0, \boldsymbol{\theta}, t)\| \leq \frac{t^{m+1}}{(m+1)!}(K_{r,i} + K_{r,g}t)e^{Kt}$$

Proof Applying Theorem 3 to the PINN learning problem for R_m, and letting R_m^* be the exact remainder obtained from the ODE solution map ψ, we obtain that

$$\|R_m - R_m^*\| \leq (K_{r,i} + K_{r,g}t)e^{Kt}$$

In turn, from Eq. (2), we obtain that $\|\varphi - \psi\| \leq \frac{t^{m+1}}{(m+1)!}(K_{r,i} + K_{r,g}t)e^{Kt}$. ∎

4. Results

In this section, we present results of running the three models against seven numerical ODE benchmark systems with varying dimensionality and number of parameters. The seven systems are as follows: a) *Duffing Oscillator*, b) *Damped Pendulum*, c) *Lorenz Attractor*, d) *Lotka-Volterra system*, e) *Rikitake Model*, f) *Susceptible-Infected-Recovered (SIR) Model* g) *Susceptible-Exposed-Infected-Recovered (SEIR) model*. Further, we show that our method scales well on higher dimensional systems. The results and prediction performance are very similar to the seven benchmarks ODEs mentioned above and their analysis can be found in Appendix C.3.

Detailed definitions for all the seven dynamical system along with the initial condition range Ω and parameter range Θ are provided in Appendix A. The PINN, Higher-Order PINN (HO-PINN), and Taylor-Model PINN (TM-PINN) are all represented using a 1 layer, 64 hidden unit shallow neural network. The input sizes vary according to the dynamical system, however, for TM-PINN we use separate neural networks to learn each dimension of the system. All the models were trained on Apple M3 Pro 14-Core GPU, running JAX 0.4.x with Metal 3 support. The seeds for all dataset creation and model initialization are provided in this codebase.
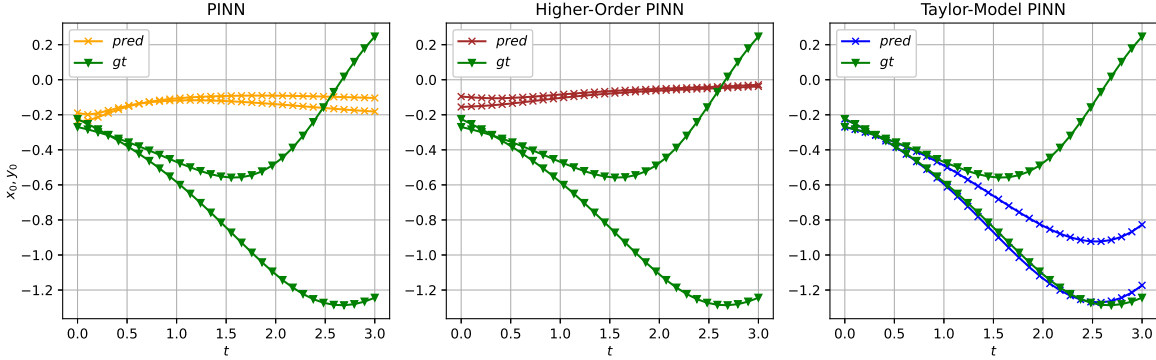

Figure 2: Prediction performance of the three models on **Duffing Oscillator** system. The initial condition are set to $x_0 = [-0.224, -0.269]$ and $\theta_0 = [0.327]$. {"gt"=ground truth, "pred"=prediction}
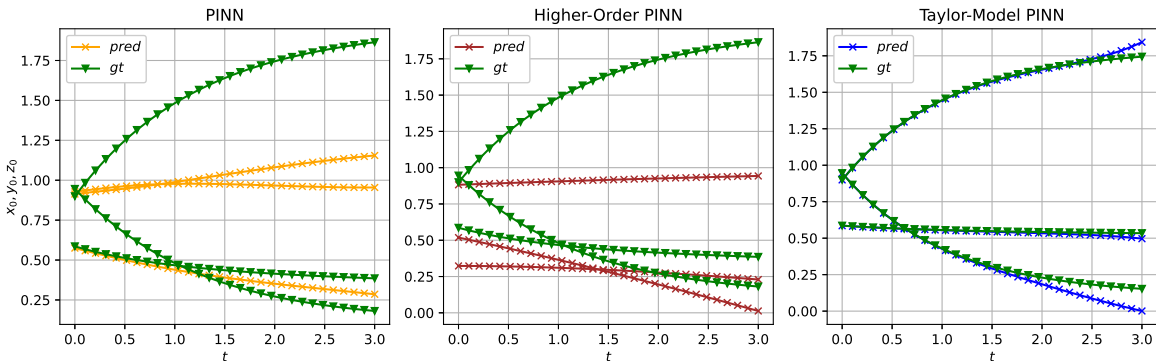

Figure 3: Prediction performance of the three models on **SIR** system. The initial conditions are set to $x_0 = [0.586, 0.945, 0.899]$ and $\theta_0 = [0.81, 0.1]$. {"gt"=ground truth, "pred"=prediction}
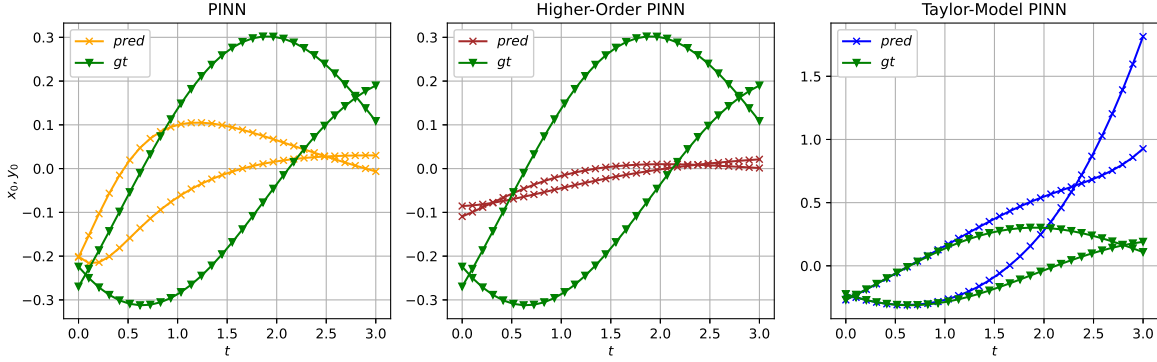
Figure 4: Prediction performance of the three models on **Damped Pendulum** system. The initial conditions and parameters are set to $x_0 = [-0.224, -0.269]$ and $\theta_0 = [0.288, 7.2]$. {"gt"=ground truth, "pred"=prediction}

4.1. Exact collocation point match for shorter time periods

We can first understand the performance of these models by looking at collocation points for shorter time windows. In Fig. 2, 3, 4 we can observe the performance of standard PINNs does not really match the collocation points after the initial few sample points. This behavior can also be observed with HO-PINNs where the initial conditions also do not tend to match. However, with further qualitative results from Tab. 1 we can observe that TM-PINNs seem to match well with the system output for larger time horizons, up to two seconds for all the benchmark systems.

Method	DO (2,1)	DP (2,2)	LV (2,4)	R (3,2)	SIR (3,2)	LoA (3,3)	SEIR (4,6)	Time (sec)
PINN	0.130	0.109	0.035	0.038	0.08	0.019	**0.069**	1
HO-PINN	0.141	0.210	0.070	0.042	0.329	0.029	246.2	1
TM-PINN	**0.003**	**0.012**	**0.004**	**0.016**	**0.002**	**0.008**	0.784	1
PINN	0.229	**0.143**	0.079	0.09	0.146	**0.022**	**0.135**	2
HO-PINN	0.24	0.215	0.133	0.09	0.334	0.035	309.2	2
TM-PINN	**0.048**	0.185	**0.06**	**0.085**	**0.023**	0.074	12.28	2
PINN	0.312	**0.16**	**0.169**	0.247	0.188	**0.025**	**0.202**	3
HO-PINN	0.329	0.21	0.251	0.248	0.354	0.038	376.4	3
TM-PINN	**0.170**	0.677	0.453	**0.206**	**0.079**	0.220	55.17	3

Table 1: Results showing (MAE↓) on Taylor-Model PINN (TM-PINN) compared against vanilla PINN and Higher-Order PINN (HO-PINN) on seven different dynamical system models across varying prediction time. {DO=Duffing Oscillator, DP=Damped Pendulum, LV=Lotka-Volterra, R=Rikitake, SIR=Susceptible-Infected-Recovered, LoA=Lorenz Attractor, SEIR=S-Exposed-IR}. The two numbers next to each model name in the header row show the number of state variables and parameters, respectively.

To understand why TM-PINNs exhibit poorer performance over longer time horizons Fig. 5, we can examine the third graph in Fig. 4. This figure shows that TM-PINNs accurately capture the evolving dynamics of the Damped Pendulum model up to approximately 1.4 seconds. However, beyond this point, the residual term fails to converge effectively, causing predictions to exceed the

expected output range of this dynamical system. This observation is further supported by the error plots and metrics presented in Appendix C. In general, we observe that TM-PINNs achieve convergence significantly faster than the other methods. However, each epoch is significantly slower due to the more complicated loss functions that involve additional terms. We find that on average across all systems, TM-PINNs require approximately 8 minutes to train for an average of 500 epochs. In comparison, PINNs take 2 minutes to complete 10^5 epochs, while HO-PINNs require 8 minutes.
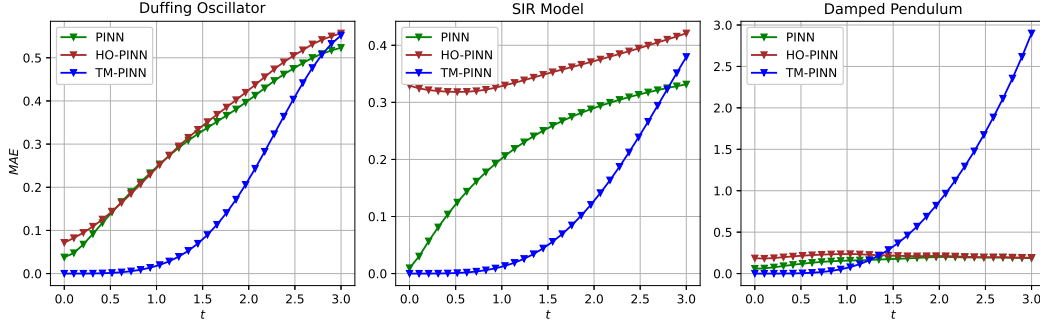
Figure 5: Avg. MAE plotted at various time points throughout the simulation of PINN, HO-PINN and TM-PNN (our approach) on three different ODEs presented in Fig. 2, 3, 4.

5. Conclusion

We have presented an approach that uses Taylor models to cast the problem of learning solutions as one for learning the remainder term from the Taylor series expansion of the solution. A theoretical analysis of our approach yields error bounds that indicate that the approach can be quite effective over smaller time horizons, while PINNs have an error growth that makes them less error-prone over longer time horizons. The extension of our work to solving PDEs using Taylor series expansions and alternatives to characterizing the remainder term remains an important part of our future work. The challenge therein lies in carefully characterizing the boundary conditions and initial conditions for the higher-order terms in the Taylor series expansion of the PDE solution. We are also interested in other types of series expansions that can approximate the ODE solutions, such as Fourier series expansions, expansions based on special functions, especially for ODEs/PDEs with oscillatory solutions Agarwal and O'Regan (2009). Specialized techniques such as power-series expansions based on Koopman operators and convergent power series expansions, especially for Lotka-Volterra-type systems, are also amenable to the approaches developed in this paper Basor and Morrison (2024).

Acknowledgments

References

Ravi P. Agarwal and Donal O'Regan. *Ordinary and Partial Differential Equations: With Special Functions, Fourier Series, and Boundary Value Problems*. Universitext. Springer New York, NY, 1 edition, 2009.

M. Althoff. An introduction to cora 2015. In *Proc. of ARCH'15*, volume 34 of *EPiC Series in Computer Science*, pages 120–151. EasyChair, 2015.

Matthias Althoff, Goran Frehse, and Antoine Girard. Set propagation techniques for reachability analysis. *Annual Review of Control, Robotics, and Autonomous Systems*, 4, 2021.

Eirini Angeloudi, Jeroen Audenaert, Micah Bowles, Benjamin M Boyd, David Chemaly, Brian Cherinka, Ioana Ciucă, Miles Cranmer, Aaron Do, Matthew Grayling, Erin E Hayes, Tom Hehir, Shirley Ho, Marc Huertas-Company, Kartheik G Iyer, Maja Jablonska, Francois Lanusse, Henry W Leung, Kaisey Mandel, Juan Rafael Martínez-Galarza, Peter Melchior, Lucas Meyer, Liam H Parker, Helen Qu, Jeff Shen, Michael J Smith, Connor Stone, Mike Walmsley, and John F Wu. The Multimodal Universe: Enabling Large-Scale Machine Learning with 100 TB of Astronomical Scientific Data. 2024.

Tom M. Apostol. *Calculus, Vol. 1: One-Variable Calculus, with an Introduction to Linear Algebra*. John Wiley & Sons, New York, 2nd edition, 1991.

David Balduzzi, Brian McWilliams, and Tony Butler-Yeoman. Neural taylor approximations: Convergence and exploration in rectifier networks, 2016.

Estelle Basor and Rebecca Morrison. Analytic solutions to nonlinear odes via spectral power series. *Linear Algebra and its Applications*, 697:561–582, Sep 2024.

Richard Bellman. The stability of solutions of linear differential equations. *Duke Math. J.*, 10(4): 643–647, 1943.

M. Berz and K. Makino. Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. *Reliable Computing*, 4:361–369, 1998.

Xin Chen and Sriram Sankaranarayanan. Reachability analysis for cyber-physical systems: Are we there yet? (invited paper). In *Proc. NASA Formal Methods Symposium*, volume 13260 of *Lecture Notes in Computer Science*, page 109–130. Springer, 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR W&CP, 2010.

E. Hairer, G. Wanner, and S. P. Nørsett. *Solving Ordinary Differential Equations I: Nonstiff Problems*. Springer, Berlin, second edition, 1993.

Youngsik Hwang and Dong-Young Lim. Dual Cone Gradient Descent for Training Physics-Informed Neural Networks, jan 2025. arXiv:2409.18426 [cs].

Fernando Iglesias-Suarez, Pierre Gentine, Breixo Solino-Fernandez, Tom Beucler, Michael Pritchard, Jakob Runge, and Veronika Eyring. Causally-Informed Deep Learning to Improve Climate Models and Projections. *Journal of Geophysical Research: Atmospheres*, 129(4):e2023JD039202, 2024. ISSN 2169-8996. doi: 10.1029/2023JD039202. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2023JD039202.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Edda Klipp, Wolfram Liebermeister, Christoph Wierling, Axel Kowald, and Ralf Herwig. *Systems Biology: A Textbook*. Wiley-VCH, 2nd edition, 2016. ISBN 9783527336364. See Chapter 5: Modeling Biochemical Reactions — examples of cascades with Michaelis–Menten steps.

S. Kong, S. Gao, W. Chen, and E. M. Clarke. dreach: δ-reachability analysis for hybrid systems. In *Proc. of TACAS'15*, volume 9035 of *LNCS*, pages 200–205. Springer, 2015.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 26548–26560. Curran Associates, Inc., 2021.

Mengfang Li, Yuanyuan Jiang, Yanzhou Zhang, and Haisheng Zhu. Medical image analysis using deep learning algorithms. *Front Public Health*, 11:1273253, nov 2023. ISSN 2296-2565. doi: 10.3389/fpubh.2023.1273253.

K. Makino and M. Berz. Rigorous integration of flows and ODEs using Taylor models. In *Proc. SNC'09*, pages 79–84, 2009.

M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, feb 2019. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Multistep neural networks for data-driven discovery of nonlinear dynamical systems, 2018.

M. G. Rosenblum, A. Pikovsky, and J. Kurths. *Synchronization – A universal concept in nonlinear sciences*. Cambridge University Press, Cambridge, 2001.

Khemraj Shukla, Patricio Clark Di Leoni, James Blackshire, Daniel Sparkman, and George Em Karniadakis. Physics-Informed Neural Network for Ultrasound Nondestructive Quantification of Surface Breaking Cracks. *J Nondestruct Eval*, 39(3):61, aug 2020. ISSN 1573-4862. doi: 10.1007/s10921-020-00705-1.

Hwijae Son, Sung Woong Cho, and Hyung Ju Hwang. Enhanced physics-informed neural networks with Augmented Lagrangian relaxation method (AL-PINNs). *Neurocomputing*, 548:126424, sep 2023. ISSN 0925-2312. doi: 10.1016/j.neucom.2023.126424.

Sophie Steger, Franz M. Rohrhofer, and Bernhard C Geiger. How PINNs cheat: Predicting chaotic motion of a double pendulum. In *The Symbiosis of Deep Learning and Differential Equations II*, 2022.

Chuwei Wang, Shanda Li, Di He, and Liwei Wang. Is L2 Physics-Informed Loss Always Suitable for Training Physics-Informed Neural Network? 2022.

Sifan Wang and Paris Perdikaris. Deep learning of free boundary and Stefan problems. *Journal of Computational Physics*, 428:109914, mar 2021. ISSN 00219991. doi: 10.1016/j.jcp.2020.109914. arXiv:2006.05311 [math].

Zixue Xiang, Wei Peng, Wen Yao, Xu Liu, and Xiaoya Zhang. Physics-informed Neural Implicit Flow neural network for parametric PDEs. *Neural Netw*, 185:107166, jan 2025. ISSN 1879-2782. doi: 10.1016/j.neunet.2025.107166.

Minglang Yin, Xiaoning Zheng, Jay D. Humphrey, and George Em Karniadakis. Non-invasive Inference of Thrombus Material Properties with Physics-informed Neural Networks. *Computer Methods in Applied Mechanics and Engineering*, 375:113603, mar 2021. ISSN 00457825. doi: 10.1016/j.cma.2020.113603. arXiv:2005.11380 [physics].

Frances Zhu, Dongheng Jing, Frederick Leve, and Silvia Ferrari. Nn-poly: Approximating common neural networks with taylor polynomials to imbue dynamical system constraints. *Frontiers in Robotics and AI*, 9, Nov 2022. doi: https://doi.org/10.3389/frobt.2022.968305.

Pavel Zwerschke, Arvid Weyrauch, Markus Götz, and Charlotte Debus. Taylor expansion in neural networks: How higher orders yield better predictions. *Iospress.nl*, page 2983–2989, 2024. doi: https://doi.org/10.3233/FAIA240838.

Appendix A. Benchmark systems

This section gives a detailed explanation of the seven benchmark ODEs that were considered for the experiments. We have mentioned the parametric and initial condition ranges that were considered for training the models. Further, we made sure to fix a unique seed value for generating all the datasets in the ranges mentioned below.

A.1. Duffing Oscillator

A nonlinear differential equation that is used to represent the dynamics of a damped oscillator is called a Duffing equation. It can be represented in two-dimensional form as follows:

$$\dot{x} = y \; ; \; \dot{y} = x - x^3 - \delta y \; ; \; (x, y) \in \Omega$$
$$\Omega \in [-0.5, 0.5] \times [-0.5, 0.5]$$

where, $\delta \in [0.1, 0.5]$ is the damping factor. The initial conditions and parameters of the model are uniform-randomly sampled from the range. The second derivatives w.r.t. time t and the Lie derivatives w.r.t the parameter of the model δ as follows:

$$\ddot{x} = x - x^3 - \delta y \; ; \; \ddot{y} = (1 - 3x^2)y - \delta(x - x^3 - \delta y)$$
$$x_\delta = 0 \; ; \; y_\delta = -y$$

A.2. Damped Pendulum

The equations of motion of a pendulum can be represented in two-dimensional form as follows:

$$\dot{\theta} = \omega \; ; \; \dot{\omega} = -b\omega - \frac{g}{L}\sin\theta \; ; \; (\theta, \omega) \in \Omega$$
$$\Omega \in [-0.5, 0.5] \times [-0.5, 0.5]$$

where, $L \in [1, 10]$ is the length of the string attached to the pendulum and $b \in [0.01, 0.5]$ is the air resistance, θ and ω are the state variables representing the angle and angular momentum of the pendulum. The initial conditions and parameters of the model are similarly sampled from the range. The second derivatives w.r.t. time t and the Lie derivatives w.r.t the parameters b, L as follows:

$$\ddot{\theta} = -\frac{g\sin\theta}{L} - b\omega \; ; \; \ddot{\omega} = -\frac{g\omega\cos\theta}{L} - b(\frac{-g\sin\theta}{L} - b\omega)$$
$$\theta_{b,L} = 0 + 0 = 0 \; ; \; \omega_{b,L} = 0 + \frac{g}{L^2}\sin\theta$$

A.3. Lotka-Volterra System

A simple biological dynamical system that describes the behavior of two species where one behaves as the predator and the other a prey is called the Lotka-Volterra system. It can be represented in two-dimensional form as follows:

$$\dot{x} = \alpha x - \beta xy \; ; \; \dot{y} = -\gamma y + \delta xy \; ; \; (x, y) \in \Omega$$
$$\Omega \in [0, 1] \times [0, 1]$$

where, $\alpha \in [0.6, 1]$ represent the per capita growth rate of the prey, $\beta \in [0.2, 0.5]$ represent the presence of predator in the prey death rate, $\gamma \in [0.5, 1.0]$ is the per capita death rate of the predator, and $\delta \in [0.1, 0.4]$ represents presence of prey in predator growth rate. Further, x, y are the state variables for the system representing the prey and predator populations, respectively. The initial condition and parameters are uniform-randomly sampled from the range. The second derivative w.r.t time t and the Lie derivatives w.r.t the parameters of the model $\alpha, \beta, \gamma, \delta$ are as follows:

$$\ddot{x} = -x(xy\delta - y\gamma)\beta + (x\alpha - xy\beta)(\alpha - y\beta) ~;~ \ddot{y} = (x\alpha - xy\beta)\gamma\delta + (xy\delta - y\gamma)(x\delta - \gamma)$$
$$x_{\alpha,\beta,\gamma,\delta} = x - xy ~;~ y_{\alpha,\beta,\gamma,\delta} = -y + xy$$

A.4. Rikitake Attractor

A dynamical system that models the behavior of a coupled magnetic dynamo, is called the Rikitake attractor. It can be simplified and represented in three-dimensional form as follows:

$$\dot{x} = -\mu x + yz ~;~ \dot{y} = -\mu y + x(z - h) ~;~ \dot{z} = 1 - xy ~;~ (x, y, z) \in \Omega$$
$$\Omega \in [-0.5, 0.5] \times [-0.5, 0.5] \times [-0.5, 0.5]$$

where, $\mu = (\omega_1 - \omega_2)\sqrt{CM/GL} \in [0.3, 0.9]$ consists of terms C, G, L, M which are the moment of inertia, applied torque, self-inductance and mutual-inductance of the dynamos, which are rotated to ω_1, ω_2 angular velocities respectively. Further, $h = R\sqrt{C/GLM} \in [0.3, 0.9]$ is another simplified term, where R is the electrical resistance. The initial condition and parameters are uniform-randomly sampled from the range. The second derivative with respect to time t and the Lie derivatives with respect to the parameter μ of the model are omitted as they get larger to be represented. One can find the symbolic derivations of these systems in the code provided.

A.5. Lorenz Attractor

A dynamical system that models atmospheric convection and exhibits chaotic behavior is called the Lorenz attractor. It can be represented in three-dimensional form as follows:

$$\dot{x} = \sigma(y - x) ~;~ \dot{y} = x(\rho - z) - y ~;~ \dot{z} = xy - \beta z ~;~ (x, y, z) \in \Omega$$
$$\Omega \in [0, 1] \times [0, 1] \times [0, 1]$$

where, $\sigma \in [0, 1]$ represent the Prandtl number, $\rho \in [0, 1]$ the Rayleigh number, and $\beta \in [0, 1]$ geometric factor. The variables x, y, z correspond to the convective flow, temperature difference, and vertical temperature variation, respectively. The initial condition and parameters are uniform-randomly sampled from the range. The second derivative with respect to time t and the Lie derivatives with respect to the parameters σ, ρ, β of the model are as follows:

$$\ddot{x} = \sigma(\dot{y} - \dot{x}) ~;~ \ddot{y} = \dot{x}(\rho - z) + x(-\dot{z}) - \dot{y} ~;~ \ddot{z} = \dot{x}y + x\dot{y} - \beta\dot{z}$$
$$x_\sigma = y - x ~;~ y_\rho = x ~;~ z_\beta = -z$$

A.6. Susceptible-Infected-Recovered (SIR) Model

The SIR model is an epidemiological framework used to describe the spread of diseases where individuals transition between being susceptible (S), infected (I), and recovered (R). It can be rep-

resented in three-dimensional form as follows:

$$\dot{S} = -\frac{IS\beta}{N} \; ; \; \dot{I} = \frac{IS\beta}{N} - I\gamma \; ; \; \dot{R} = I\gamma \; ; \quad (S, I, R) \in \Omega$$
$$\Omega \in [0, 1] \times [0, 1] \times [0, 1]$$

where, $\beta \in [0, 1]$ is the probability of disease transmission per contact, $\gamma \in [0, 1]$ is the per-capita recovery rate. The initial conditions and parameters of the model are uniform-randomly sampled from the range. The second derivatives w.r.t time t and the Lie derivatives w.r.t. the parameters β, γ are as follows:

$$\ddot{S} = \frac{KS\beta}{N} + L \; ; \; \ddot{I} = -L + \frac{K(-N\gamma + S\beta)}{N} \; ; \; \ddot{R} = K\gamma$$
$$S_{\beta,\gamma} = -\frac{SI}{N} \; ; \; I_{\beta,\gamma} = \frac{SI}{N} - I \; ; \; R_{\beta,\gamma} = I$$

where, $K = \frac{IS\beta}{N} - I\gamma$, $L = \frac{\beta^2 I^2 S}{N^2}$

A.7. Susceptible-Exposed-Infected-Recovered (SEIR) Model

A compartmental epidemiological model that describes the spread of infectious diseases with temporary immunity is called the SEIR model. It can be represented in four-dimensional form as follows:

$$\dot{S} = \mu(N - S) - \frac{\beta SI}{N} + \omega R \; ; \; \dot{E} = \frac{\beta SI}{N} - (\sigma + \mu)E \; ;$$
$$\dot{I} = \sigma E - (\mu + \gamma + \alpha)I \; ; \; \dot{R} = \gamma I - (\mu + \omega)R \; ; \; S, E, I, R \in \Omega$$
$$\Omega \in [0, 0.99] \times [0, 0.99] \times [0, 0.5] \times [0, 0.5]$$

where, $\mu \in [0.01, 0.02]$ represent the birth/death rate, $\beta \in [0.01, 0.02]$ transmission rate, $\sigma \in [0.1, 0.2]$ is the incubation rate, $\gamma \in [0.01, 0.2]$ is the recovery rate, $\alpha \in [0, 0.5]$ is the disease-induced death rate, and $\omega \in [0.1, 1]$ is the loss of immunity rate. Further, the variables S, E, I, R correspond to the susceptible, exposed, infected, and recovered populations, respectively. The initial condition and parameters are uniform-randomly sampled from the range. The second derivative with respect to time t and the Lie derivatives with respect to the parameter μ of the model are omitted as they get larger to be represented.

Appendix B. Numerical Quadrature insights

To support our final approach of learning the remainder term, we proposed the numerical quadrature method. In this section, we define and compute the method against three systems A.1, A.2, A.3 to showcase how approximation using quadratures affect the performance. We set the number of quadrature points to 10 and obtain the remainder term R_m as follows:

$$R_m(\boldsymbol{x}_0, \boldsymbol{\theta}, t) \approx \frac{4}{10} \left(\frac{1}{2}F(0) + F(0.1) + F(0.2) + \cdots + \frac{1}{2}F(1) \right)$$

The results shown in Tab. 2 indicate that for smaller number of quadrature points, the error increases rapidly over time making the predictions highly erroneous across all systems. However, since the

Method	DO (2,1)	DP (2,2)	LV (2,4)	Time (sec)
TM-PINN	0.003	**0.012**	0.004	1
TM-PINN-NQ	**0.002**	0.016	**0.002**	1
TM-PINN	0.048	**0.185**	**0.06**	2
TM-PINN-NQ	**0.046**	0.477	0.077	2
TM-PINN	**0.170**	**0.677**	**0.453**	3
TM-PINN-NQ	0.218	4.155	0.665	3

Table 2: Results showing (MAE↓) on Taylor-Model PINN using 10 Numerical Quadrature points (TM-PINN-NQ) compared against TM-PINNs as reported in the main paper on three different dynamical systems. {DO=Duffing Oscillator, DP=Damped Pendulum, LV=Lotka-Volterra}.
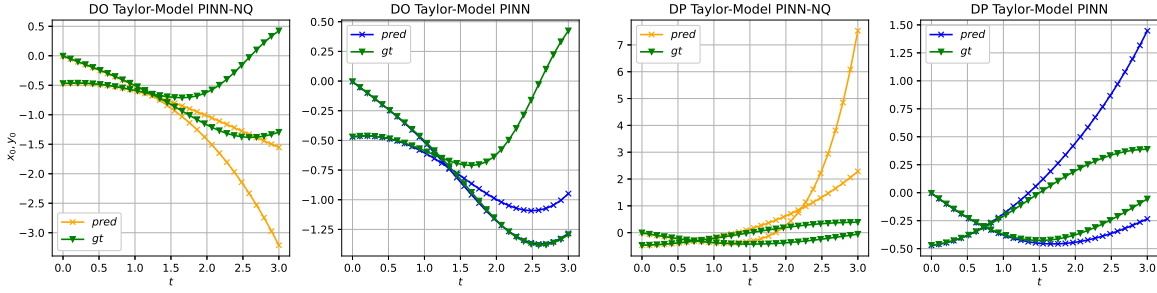

Figure 6: Prediction performance of the both models Taylor-Model PINN using 10 Numerical Quadrature points (TM-PINN-NQ) and TM-PINN, on {DO=Duffing Oscillator} and {DP=Damped Pendulum} systems. The initial conditions of the dynamical system are set as before.

number of points are small the algorithm takes less time to train the neural network as there is no longer expensive differential computation at each epoch. Further, Fig. 6 shows the prediction performance of training the neural network using both the approaches on the first two dynamical system (we see similar performance for the Lotka-Volterra system as well).

Appendix C. Supporting results from other systems

In this section, we continue with the results from our experiments on the benchmark systems. First, we can continue looking at the prediction plots for the rest of the systems. Secondly, we look at the error propagation through time for the various models. Finally, we look at scaling the systems to larger dimensional ODEs and study the predictive performance in this scenario.

C.1. Prediction performance graphs

The following Fig. 7, 8, 9, 10 show the prediction performance of the three models across the same set of initial condition and parameter space.

C.2. Results and error propagation

In Fig. 11 we can further see that across all the remaining four systems, as noted earlier, TM-PINNs have negligent mean absolute error across the first few seconds of the evolution. But, the error tends
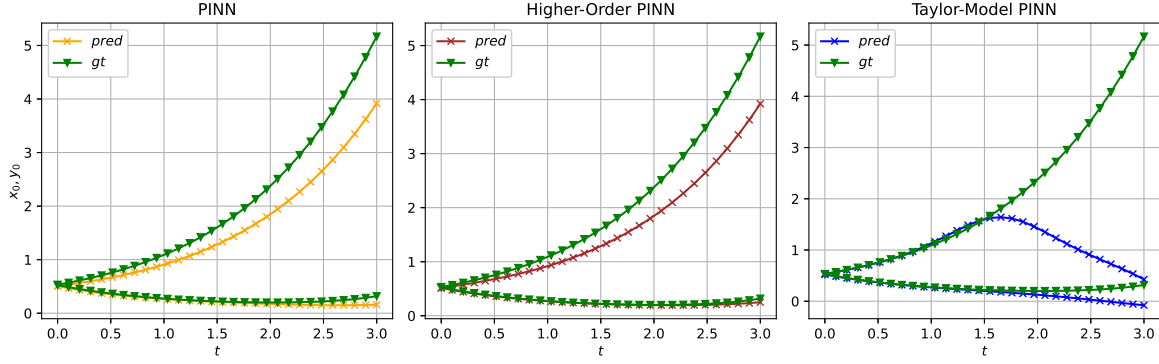
Figure 7: Prediction performance of the three models on **Lotka-Volterra** system. The initial conditions are set to $\boldsymbol{x}_0 = [0.53, 0.52]$ and $\boldsymbol{\theta}_0 = [0.87, 0.43, 0.95, 0.39]$
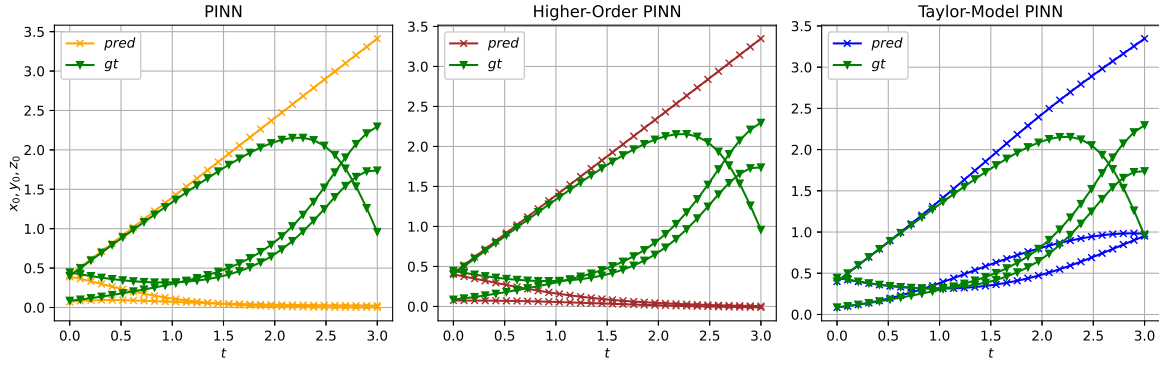

Figure 8: Prediction performance of the three models on **Rikitake** system. The initial conditions of the dynamical system are set to $\boldsymbol{x}_0 = [0.08, 0.44, 0.39]$ and $\boldsymbol{\theta}_0 = [0.49, 0.69]$
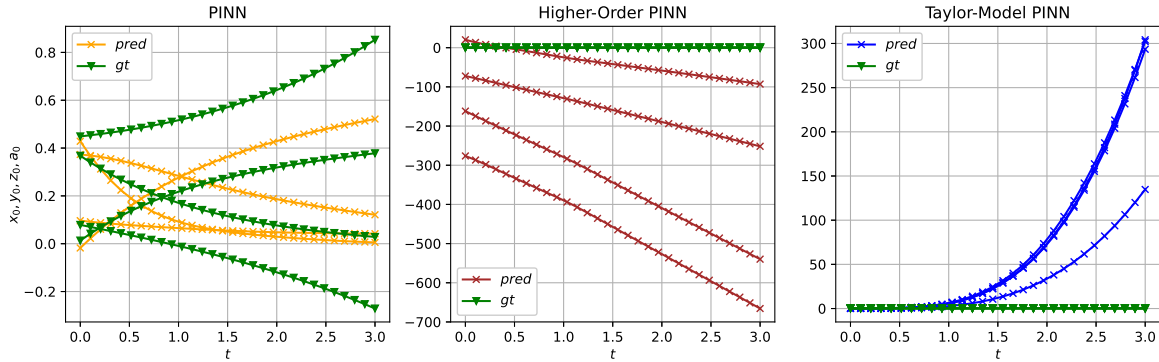

Figure 9: Prediction performance of the three models on **SEIRS** system. The initial conditions are set to $\boldsymbol{x}_0 = [0.01, 0.078, 0.45, 0.36]$ and $\boldsymbol{\theta}_0 = [0.01, 0.2, 0.17, 0.76, 0.36, 0.049]$

to grow as prediction time horizon increases. Finally, we also compute the Root Mean Square Error (RMSE) and notice a similar performance across models Tab. 3.
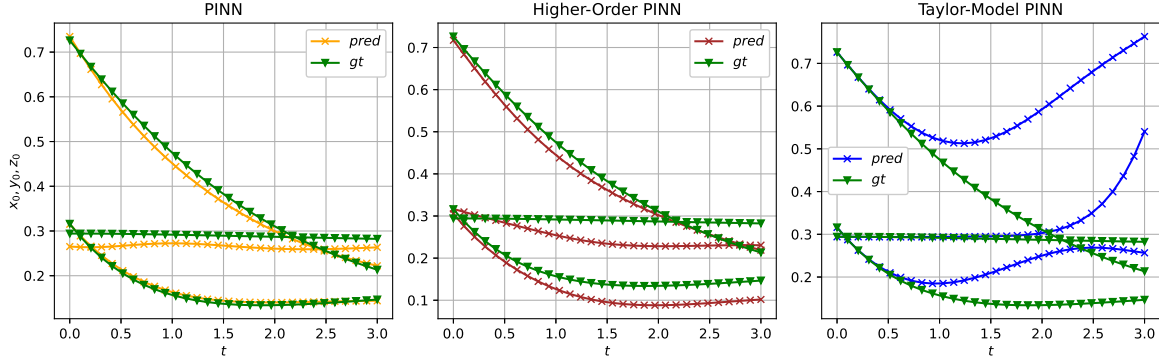
18

Figure 10: Prediction performance of the three models on **Lorenz Attractor** system. The initial conditions of the dynamical system are set to $\boldsymbol{x}_0 = [0.29, 0.32, 0.73]$ and $\boldsymbol{\theta}_0 = [0.53, 0.03, 0.79]$
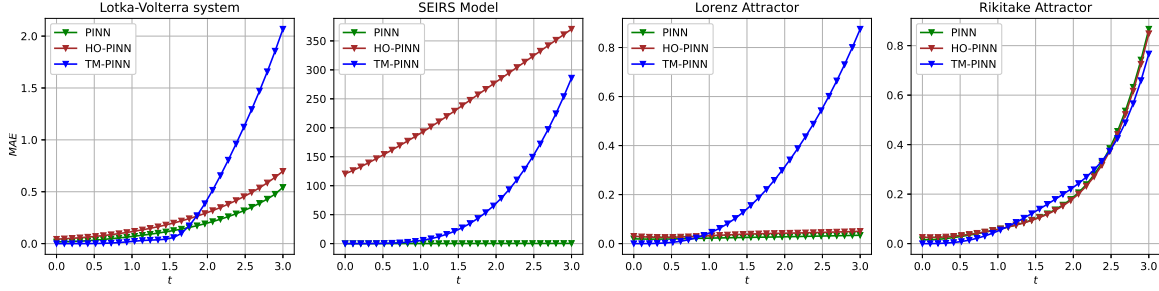

Figure 11: MAE plotted at various time points the three different models comparing the PINN, HO-PINN and the TM-PNN (our approach).

Method	DO (2,1)	DP (2,2)	LV (2,4)	R (3,2)	SIR (3,2)	LoA (3,3)	SEIR (4,6)	Time (sec)
PINN	0.178	0.154	0.051	0.055	0.120	0.027	**0.102**	1
HO-PINN	0.186	0.264	0.094	0.055	0.399	0.039	253.7	1
TM-PINN	**0.008**	**0.046**	**0.007**	**0.028**	**0.004**	**0.021**	1.346	1
PINN	0.307	**0.195**	0.136	0.149	0.202	**0.033**	**0.198**	2
HO-PINN	0.321	0.267	0.210	0.145	0.409	0.050	321.901	2
TM-PINN	**0.119**	0.664	**0.123**	**0.142**	**0.045**	0.167	20.78	2
PINN	0.411	**0.214**	**0.349**	0.469	0.255	**0.040**	**0.299**	3
HO-PINN	0.433	0.262	0.481	0.468	0.431	0.059	396.9	3
TM-PINN	**0.337**	2.130	0.912	**0.371**	**0.150**	0.478	92.44	3

Table 3: Results showing (RMSE↓) on TM-PINN compared against vanilla PINN and HO-PINN on seven different dynamical system models across varying prediction time. {DO=Duffing Oscillator, DP=Damped Pendulum, LV=Lotka-Volterra, R=Rikitake, SIR=Susceptible-Infected-Recovered, LoA=Lorenz Attractor, SEIR=S-Exposed-IR }.

19

C.3. Larger systems

Addressing the reviews, we ran our method against two larger systems. (a) A multi-coupled damped oscillator, building upon the system provided in Rosenblum et al. (2001), (b) Michaelis-Menten kinetics system similar to the system provided in Klipp et al. (2016).

C.3.1. MULTI-COUPLED DAMPED OSCILLATOR

The multi-coupled damped oscillator is a 8-dimensional nonlinear differential equation similar to the damped oscillator model used in our benchmarks (Appendix A) and can be represented as follows:

$$\dot{x}_1 = y_1 \; ; \; \dot{y}_1 = \mu(1 - x_1^2)y_1 - x_1 + \delta(x_2 - 2x_1 + x_4)$$
$$\dot{x}_2 = y_2 \; ; \; \dot{y}_2 = \mu(1 - x_2^2)y_2 - x_2 + \delta(x_3 - 2x_2 + x_1)$$
$$\dot{x}_3 = y_3 \; ; \; \dot{y}_3 = \mu(1 - x_3^2)y_3 - x_3 + \delta(x_4 - 2x_3 + x_2)$$
$$\dot{x}_4 = y_4 \; ; \; \dot{y}_4 = \mu(1 - x_4^2)y_4 - x_4 + \delta(x_1 - 2x_4 + x_3)$$

where, $\{x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4\} \in [-0.5, 0.5]$ are the state variables of the system and $\delta, \mu \in [0.1, 0.5]$ are the damping factors of the coupled system. The initial conditions and parameters of the model are uniform-randomly sampled from the range. The second derivatives w.r.t. time t and the Lie derivatives w.r.t the parameter of the model δ, μ are symbolically created.

C.3.2. MICHAELIS-MENTON ENZYME KINETICS

The Michaelis-Menten system explains how presence of some i-enzyme concentrations in an enzyme-substrate complex can cause kinetic rate enhancement of a reaction. We extend this system to 6-dimensions i.e. six enzymes present in the concentrate, which can be represented as follows:

$$\dot{x}_1 = \frac{V_1}{K_m + 1} - \delta x_1 \; ; \; \dot{x}_2 = \frac{V_2 x_1}{K_m + x_1} - \delta x_2$$
$$\dot{x}_3 = \frac{V_3 x_2}{K_m + x_2} - \delta x_3 \; ; \; \dot{x}_4 = \frac{V_4 x_3}{K_m + x_3} - \delta x_4$$
$$\dot{x}_5 = \frac{V_5 x_4}{K_m + x_4} - \delta x_5 \; ; \; \dot{x}_6 = \frac{V_6 x_5}{K_m + x_5} - \delta x_6$$

where, $\{x_1, x_2, x_3, x_4, x_5, x_6\} \in [0.1, 0.5]$ are the state variables and $\{V_1, V_2, V_3, V_4, V_5, V_6\} \in [0.5, 1.0]$ are the maximum reaction velocities for each enzyme present in the substrate. We set the Michaelis constant $K_m = 0.5$ and the degradation rate constant $\delta = 0.1$. The initial conditions and parameters of the model are uniform-randomly sampled from the range. The second derivatives w.r.t. time t and the Lie derivatives w.r.t the parameter of the model V_i are symbolically created.

To this end, we run all three models against the above two systems. The hyperparamters of the models are kept the same as other experiments reported, however, the learning rate is reduced to 0.005 (high dimensional systems have sharper gradients and many local minima) and the time duration T is reduced to 2 seconds with same intervals (to accomodate hardware limitations). We notice similar performance to previous methods where TM-PINNs perform well on shorter time horizons compared to other methods. Tab. 4 gives us the MAE across each system and Fig. 13, 12 show the prediction performance of the models.

Method	MMEK (6,6)	CDO (8,2)	Time (sec)
PINN	0.020	0.061	1
HO-PINN	0.157	0.188	1
TM-PINN	**0.005**	**0.005**	1
PINN	**0.030**	**0.112**	2
HO-PINN	0.171	0.220	2
TM-PINN	0.097	0.130	2

Table 4: Results showing MAE(↓) on TM-PINN compared against vanilla PINN and HO-PINN on two larger dynamical system models across varying prediction time. {MMEK=Michaelis-Menton Enzyme Kinematics, CDO=Coupled Damped Oscillators}
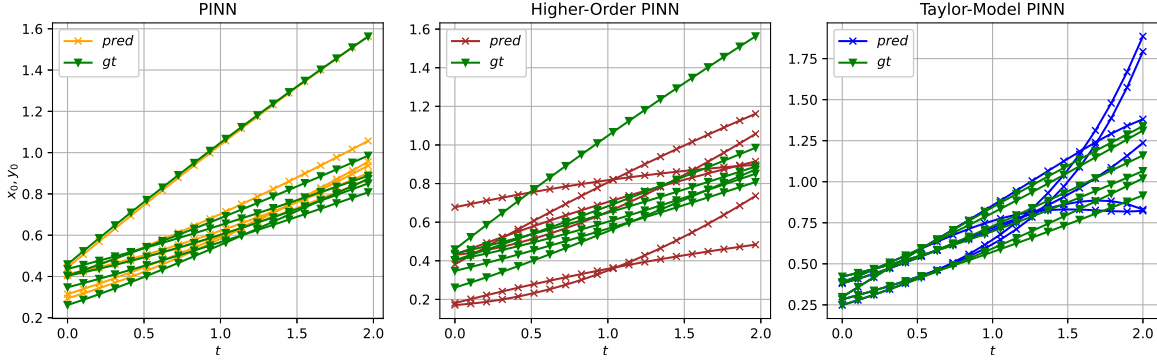

Figure 12: Prediction performance of the three models on **Michaelis-Menton Enzyme Kinematics** system. The initial conditions of the dynamical system are set to $\boldsymbol{x}_0 = [0.29, 0.38, 0.38, 0.28, 0.24, 0.42]$ and $\boldsymbol{\theta}_0 = [0.9, 0.76, 0.93, 0.62, 0.85, 0.74]$
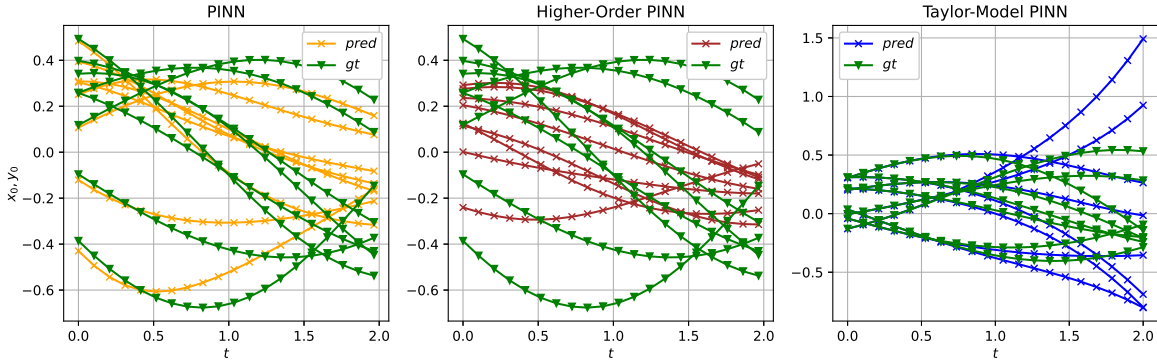

Figure 13: Prediction performance of the three models on **Coupled Damped Oscillator** system. The initial conditions of the dynamical system are set to $\boldsymbol{x}_0 = [0.00, 0.20, 0.21, -0.04, -0.13, 0.3, 0.31, 0.03]$ and $\boldsymbol{\theta}_0 = [0.44, 0.19]$

Appendix D. Algorithm for Taylor-Model PINNs

In this section, we provide the algorithm for training TM-PINNs as used in our experiments. We start by symbolically computing the m Lie derivatives of the dynamical system f provided, and use a neural network ψ to numerically approximate the remainder term. The loss functions, defined in 3.1 is implemented as shown in Alg. 1 pseudocode.

Algorithm 1 Taylor Model Physics-Informed Neural Networks (TM-PINNs)

1: **Require:** Training data $\mathcal{D} = (\boldsymbol{x}_i^0, \boldsymbol{\theta}_i^0, \boldsymbol{t})_{i=1}^N$ where $\boldsymbol{x}_0$ and $\boldsymbol{\theta}_0$ are the initial conditions and parameters, $\boldsymbol{t}$ represents the time horizon up to time T divided by Δt, number of epochs N_{iter}, the dynamical system $f(\boldsymbol{x}_0, \boldsymbol{\theta}_0, t)$, and a neural network ψ with weights and biases $(\boldsymbol{w}, \boldsymbol{b})$.

2: **Initialize:** Symbolically compute the m Lie derivatives $\mathcal{L}^{(m)}$ of the system f

3: **Training:**

4: **for** $i = 1$ to N_{iter} **do**

5:     Sample a random mini-batch of training data $\boldsymbol{b} \subset \mathcal{D}$

6:     $\boldsymbol{g}_r \leftarrow \boldsymbol{b} + t\mathcal{L}^{(1)}(\boldsymbol{b}) + \cdots + \frac{t^m}{m!}\mathcal{L}^{(m)}(\boldsymbol{b})$

7:     $\boldsymbol{g}_l \leftarrow \mathcal{L}^{(1)}(\boldsymbol{b}) + t\mathcal{L}^{(2)}(\boldsymbol{b}) + \cdots + \frac{t^{(m-1)}}{(m-1)!}\mathcal{L}^{(m)}(\boldsymbol{b})$

8:     $\hat{\boldsymbol{b}}, \dot{\boldsymbol{b}} \leftarrow \psi(\boldsymbol{b}), \nabla\psi(\boldsymbol{b})$

9:     $\boldsymbol{g}_r \leftarrow f(\boldsymbol{g}_r + \frac{t^{(m+1)}}{(m+1)!}\hat{\boldsymbol{b}}, \boldsymbol{\theta}_0, t)$

10:     $\boldsymbol{g}_l \leftarrow \boldsymbol{g}_l + \frac{t^m}{m!}\hat{\boldsymbol{b}} + \frac{t^{(m+1)}}{(m+1)!}\dot{\boldsymbol{b}}$

11:     $\nabla_{\boldsymbol{w},\boldsymbol{b}}\mathcal{L}(\boldsymbol{w}, \boldsymbol{b}) \leftarrow \nabla_{\boldsymbol{w},\boldsymbol{b}}(||\boldsymbol{g}_l - \boldsymbol{g}_r||_2^2) + \nabla_{\boldsymbol{w},\boldsymbol{b}}(||\mathcal{L}^{(m+1)}(\boldsymbol{b}) - \hat{\boldsymbol{b}}||_2^2)$

12:     Update weights of ψ using ADAM

13: **end for**

Here, we assume the same gradient update steps as used by the Adaptive Moment Estimation (ADAM) optimizer Kingma and Ba (2014), with some learning rate η. All the neural network weights are initalized using the Glorot initialization Glorot and Bengio (2010) i.e. sampled from a uniform random distribution of mean 0 and variance $\sqrt{\frac{2}{\xi_{in}+\xi_{out}}}$, where ξ_{in} in the number of input layers and ξ_{out} is the number of output layers.