PHYSICS-INFORMED DEEP OPERATOR CONTROL: Controlling chaos in van der Pol oscillating circuits

Hanfeng Zhai & Timothy Sands*

Sibley School of Mechanical and Aerospace Engineering,
Cornell University, Ithaca, NY

December 25, 2021

Abstract

Controlling chaos is a long-standing problem in engineering. By linearizing nonlinear systems, traditional control methods cannot learn features from chaotic data for control. Here, we introduce Physics-Informed Deep Operator Control (PIDOC), by encoding control signal and initial position into the losses of a Physics-Informed Neural Network (PINN), the nonlinear system exhibits the desired route given the control signal. The framework has a clear application scenario: receiving signals as physics commands and learning chaotic data from the nonlinear van der Pol system, PIDOC is able to execute the controlled signal as the output of the PINN. PIDOC is first applied to a benchmark problem, unveils a fluctuating behavior of the acceleration that seemingly behave in the same frequency of the desired signal, indicating the learning of neural networks (NN) elicit stochasticity for higher-order terms, yet still successfully impose the control that enforces the system behave as in the same frequency of the control signal. PIDOC is also proved capable of converging to different desired trajectories based on case studies. Initial positions slightly affect the control accuracy at the beginning stage yet still converge to the control signal based on numerical experiments. To note, for highly nonlinear systems, PIDOC is not able to execute control as high accuracies compared with the benchmark problem, yet still, PIDOC converges to decent trajectories corresponding to given signals with fair predictable behavior. The depth and width of the NN structure did not heavily variate the convergence of PIDOC based on case studies on van der Pol systems with low and high nonlinearities. Surprisingly, by enlarging the control signal by multiplying a Lagrangian multiplier, PIDOC failed to converge to the desired trajectory, indicating enlarging the control signal does not increase control quality, which is not intuitive. The proposed framework can be potentially applied to many nonlinear systems for controlling chaos.

Keywords: Physics-informed neural networks; van der Pol dynamics; nonlinear control; chaos

1 Introduction

CONTROLLING nonlinear dynamics and chaos is a long-standing issue in various engineering disciplines including aerospace systems design [1], chemical operations [2], robotics [3], biological sciences [4], mechatronics [5], and specifically, microelectronics [6], especially for circuits systems involving semiconductor that elicit nonlinearity for signal controls [7, 8]. In mathematics, chaos is defined as a scenario that typical solutions of a differential equation (or a representation

*To whom correspondence should be addressed. Email: tas297@cornell.edu

of the system) do not converge to a stationary or periodic function of time but continue to exhibit a seemingly unpredictable behavior [9]. Controlling chaos has a simple objective: implementing the desired command to the system to make the system behave "as we wish" or at least to make the system predictable so human impotence can be executed. However, controlling chaos is arduous [10], yet ubiquitous in nature as in fluid flows, heartbeat irregularities, weather, and climate [11, 12, 13]. Hence, solving such a problem is crucial.

Traditionally, proportional–integral–derivative controllers (PID) were widely adopted for controlling nonlinear systems [15, 16], which employ closed-loop feedback errors that can tune feedforward command as close to the target output as possible[17], are still common in nowadays's industry. Particularly, a common practice in control theory is to implement trigonometric command as for predictable periodic system behavior [18]. Notably, with such wide applications, PID control experiences sluggish and systematic postpone due to the integral term, and order increasing might lead to system instability [19]. Admittedly, reducing PID to PI (proportional-integral) or P (proportional) might either increase speed or system stability, but neither can learn the features of nonlinear systematic data, which allows more advanced self-adjust control behavior.

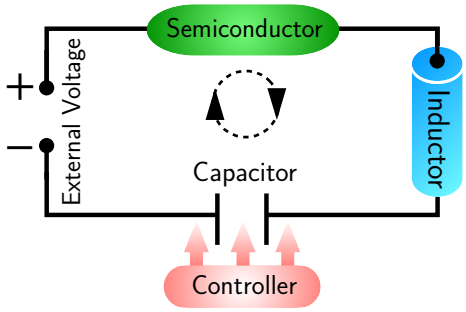

Figure 1: The schematic diagram for the van der Pol circuits. Note that the nonlinearity origins from the semiconductor (the green part), cause the system to generate chaotic behavior. The cycle of the oscillating circuits generates current as illustrated by the dashed circle. See text for details.

In the past decades, the skyrocketing big data, assisted with advanced computing technologies such as GPU computing [21], incited the development of machine learning algorithms, specifically, deep neural networks [20], as they can learn and capture features from highly nonlinear data for accurate predictions, showing huge potentials that attracted public's attention in various fields. In recent years, utilizing physics information to encode in the losses of a deep neural network (NN), promising faster accurate learning of physics with NN that respect basic physics laws with less labeled data, commonly recognized as Physics-Informed Neural Networks (PINNs) [22, 23]. One of the celebrated characteristics of PINNs is they can learn from sparse data [24] as physics doesn't generate humongous data as easily in other commercial fields. Upon the proposed PINN framework, various types of PINNs designed for different engineering applications emerge in fields, with their most renowned works in predicting fluid fields [25, 26], but also include electronics [28, 29]. Henceforth a question arose: can PINN be applied for dynamical systems? Notably, there are a few attempts using the PINN-based method to learn and predict nonlinear dynamical systems and chaos [31, 32, 33, 34]; with a notable good attempt on using it to learn and map the van der Pol system [35]. Notwithstanding, the applications mostly focus on "learn" and "discovery" of dynamics with PINNs, with few actually focusing on "controlling" the system, that is to guide the system to behave as human-desired signals. Also, a few good attempts address NN for controls [36, 37, 38], as limited to replacing a block or parts of the closed-loop framework with a NN, rather

than directly applying NN-based framework for signal controls.

Acknowledging the limitations of PINNs and other NN-based controls, a question thence emanate: can control signals be incorporated in PINNs as for nonlinear dynamical systems? To investigate such, we focus on a van der Pol system, proposed by van der Pol in the early 20th century while studying oscillating circuits [39, 40, 41]. The van der Pol system exhibits highly chaotic behavior that encompasses wide applications in biology, biochemistry, and microelectronics [42, 43, 44]. Traditional control on the van der Pol system usually includes linearizing the system or adding a forced term to impose control [45, 46, 47, 53]. A basic setup of the oscillating van der Pol circuit is illustrated in Figure 1: an external voltage excites the circuit that induces a current, which can be converted to a charge through the capacitor [48]. If the semiconductor as indicated in green is equipped, the circuit will display highly nonlinear behavior as hard to control and predict. One can design a controller as indicated red block for controlling chaos in circuits, is our main goal.

Inspired by PINNs [22], and focusing on the van der Pol system as different control methods strives to control [53], we proposed Physics-Informed Deep Operator Control (PIDOC), a PINN-based control method that incorporates the generation losses of NN, the desired control signal, and the initial position of the system into the loss function of a PINN that are able to output the desired control signal. The framework is tested based on its behavior and its ability to deal with the highly nonlinear system; the hyperparameters are also investigated for better interpretation and potential applications of PIDOC.

The paper is arranged as follows: in Section 2 we first formulate the problem of a van der Pol system and controls (Sec. 2.1), and elaborate the basic system setup in Section 2.2. In Section 3 the detailed methodologies of formulating and the learning of PIDOC is described, consisting basis of deep learning (Sec. 3.1) and physics-informed control (Sec. 3.2). It is briefly summarized on how the numerical experiments are conducted in Section 3.3. Followed by Section 4 showing the results and discussion on PIDOC: Section 4.1 analyze the behavior of PIDOC given a benchmark problem; followed by an in-depth estimation of nonlinearity of trajectory convergence in Section 4.2, estimated the amplitude of control signals (Sec. 4.2.1), influences of initial positions (Sec. 4.2.2), and noninearity analysis (Sec. 4.2.3). The hyperparameters of the NN (Sec. 4.3.1) and the weight of the control signal in PINN (Sec. 4.3.2) is further studied in Section 4.3. The paper is eventually concluded and summarized in Section 5.

2 Problem Formulation

The nonlinear control of the chaotic van der Pol system problem is mainly inspired by the work of Cooper *et al.* [53], with a clear problem objective: the effort of a successful implementation of the desired trajectory to the van der Pol system.

2.1 van der Pol dynamics

van der Pol describes the oscillatory behavior to the class of nonlinear equations that are now referred to the van der Pol equation [39], where originally proposed to describe oscillating circuits with semiconductors as introduced in Section 1, writes:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = 0 \tag{1}$$

where if $x(t)$ is referred as position, thence $\dot{x}(t)$ is the velocity and $\ddot{x}(t)$ is the acceleration; μ is a scalar parameter indicating the nonlinearity and the strength of the damping.

The equation exhibits an oscillatory behavior, yet with a non-constant amplitude, representing an invariant trajectory set called a "limit cycle" [53]. System trajectories converge to these invariant orbits given any initial conditions, as can be referred to in Figure 2: Given six different picked initial points, all converge to the non-constant limit cycle, with a changing velocity to positions, indicating a non-stable phase. The robustness of the nonlinear phase suffers scientists as the system is automatically "trapped" in the state as long as the control signals cease, indicating the robustness of the inherent dynamics.

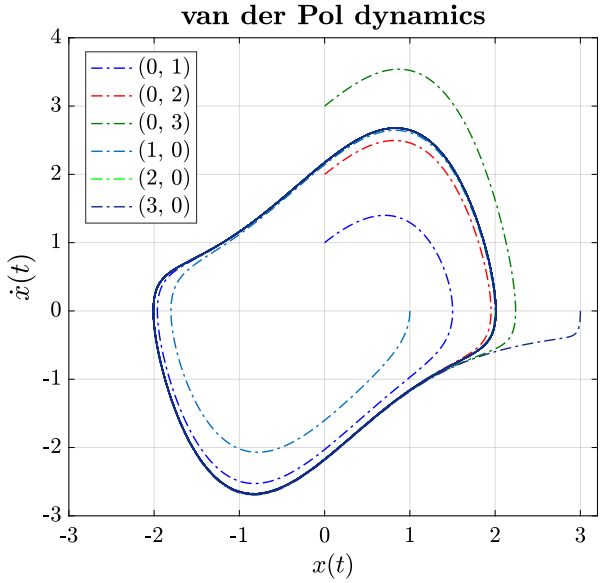

Figure 2: The phase portrait for the nonlinear van der Pol equation when $\mu = 1$. The phase is initiated from six different points: $(1,0)$, $(2,0)$, $(3,0)$, $(0,1)$, $(0,2)$, $(0,3)$, as indited in different dashed lines in the box, both converge to the same nonlinear trajectory of the van der Pol inherent dynamics. Note that the data for this figure is generated using `odeint` in `SciPy` library.

As indicated in Section 1, the main goal addressed is to control such nonlinear behavior. Seeking to produce a fixed-amplitude oscillation, traditional control methods commonly add forcing functions to the nonlinear equation as in Equation (2), which allows linear time-invariant (LTI) feedback controller based on a linearized version of the system equation, as for classical control design [53]:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2)\frac{dx}{dt} + x = F(t) \tag{2}$$

However, in this paper such an attempt is not adopted: the goal is to control the unsteady state of oscillating dynamics by learning from the original van der Pol equation receiving an external control signal to the controller, with no external terms added to modify the original van der Pol formulation. Such an attempt won't suffer from the errors of linearization as in traditional methods such as PI or PID controllers [14, 15, 16], neither does it accumulate the errors from integration as in PID controls. What's more, the NN-based control PIDOC is able to capture the feature from nonlinear chaotic data directly from the van der Pol system (through the capacitor in Figure 1); and the optimization of minimizing losses is an innate feedback process to the framework for control as the errors are reduced per iterations, with the feed-forward control signal is output as the

predictions from the learning of the NN. For system controls, Input a sinusoidal or other triangular functions through modifying $F(t)$ in Equation (2) is a classical approach; where we incorporate such a signal as an external supervision [30] added to PIDOC in the paper.

2.2 System simulation

Depicting and emulate the system as described in Figure 1 requires a simulation of van der Pol system that generate nonlinear data, which are fed into PIDOC for control. In the classical definition of a NN, such data are considered as training data, for the NN to "learn" and "predict". Here, the `Python` differential equation solver `odeint`, part of `SciPy` library, is adopted to solve van der Pol equation for system simulation [49]. The `odeint` is a library containing advanced numerical methods for solving differential equations, especially for initial value problems [50]. Compare with other solver such as `scipy.integrate.solve_ivp`, `odeint` are able to generate data with higher smoothabilities [51], promising a "cleaner" data fed for NN training and learning. In this paper, the van der Pol equation is solved for $t = 30$, interpolated with 3000 points, as in Equation (1). The parameters determine the error control performed by the solver `rtol` and `atol` is chosen as 10^{-6} and 10^{-10}, respectively [49].

3 Methodology and Algorithm

In this section, we will introduce the basic setup of PIDOC, employing PINN as initial position (denoted by $\mathcal{I}$) and control signal (denoted by $\mathcal{D}$, meaning desired trajectory) encoded within the NN losses for accurate controls.

3.1 Deep learning

PIDOC controls nonlinear systems based on "learning" from chaotic data for output a prediction as control, which are enabled by a deep neural network (DNN). For the van der Pol system, the DNN is formulated as:

$$x_{pred} = (K_L \circ \sigma_L \circ ... \circ K_1 \circ \sigma_1 \circ K_0)t \tag{3}$$

where the DNN outputs an desired trajectory x_{pred} given an input of the specific time series t. $K_1, K_2, ..., K_L$, are linear layers; $\sigma_1, \sigma_2, ..., \sigma_L$ are the activation functions, where PIDOC employs `tanh` activation functions. For PIDOC, the NN take time t as the input though layer K_0 to transmit through $(L-1)^{th}$ hidden layers to generate an output x_{pred} though the output layer K_L, supervised from the chaotic data of the van der Pol system x_{train}. Here, Let $\mathcal{N}^L \equiv (K_L \circ \sigma_L \circ ... \circ K_1 \circ \sigma_1 \circ K_0)$ denotes the L^{th} layers NN.

The DNN corresponds to Equation (3) are commonly recognized as a combination of three parts: input, hidden, and output layers, where the neurons are connected, commonly known as a feed-forward NN, defined recursively as

$$\begin{aligned} \text{Input layer}: \ & K_0(t) = t \subset \mathbb{R}^{d_{in}} \\ \text{Hidden layer}: \ & K_{\mathfrak{L}}(t) = \sigma\left(\mathbf{w}_{\mathfrak{L}} K_{\mathfrak{L}-1}(t) + \mathbf{b}_{\mathfrak{L}}\right) \subset \mathbb{R}^{\mathcal{K}^{\mathfrak{L}}}, \text{ for } 1 \leq \mathfrak{L} \leq L-1 \\ \text{Output layer}: \ & K_L(t) = \mathbf{w}_L K_{L-1} + \mathbf{b}_L \subset \mathbb{R}^{d_{out}}, \ \left(x_{pred} \equiv K_L(t)\right) \end{aligned} \tag{4}$$

For the NN $\mathcal{N}^L(t) : \mathbb{R}^{d_{in}} \longrightarrow \mathbb{R}^{d_{out}}$, denotes the input time series t is transmit to the linear input layer K_0; forward through hidden layers, from a linear model with $\mathbf{w}_{\mathfrak{L}}$ as weights and $\mathbf{b}_{\mathfrak{L}}$ as biases

activated through activation function `tanh`, to generate an output through K_L for NN predictions x_{pred}, can also interpreted as the *controlled dynamics*. To be videlicet, there are $\mathcal{K}^{\mathfrak{L}}$ neurons in the $\mathfrak{L}^{th}$ layer $\left(\mathcal{K}^0 = d_{in}\ \&\ \mathcal{K}^L = d_{out}\right)$, the weights and biases are thence denoted by $\mathbf{w}_{\mathfrak{L}} \subset \mathbb{R}^{\mathcal{K}^{\mathfrak{L}} \times \mathcal{K}^{\mathfrak{L}-1}}$ and $\mathbf{b}_{\mathfrak{L}} \subset \mathbb{R}^{\mathcal{K}^{\mathfrak{L}}}$ [27]. Note that the visualization of the NN is shown in the blue box in Figure 3.

In the PIDOC formulation, a supervised machine learning problem is defined, where the NN learning is enabled through the supervision of external training data, as a formulation on minimizing the loss function, so that the NN can capture data structures through this optimization process, where in traditional NN aproaches $\mathcal{L}$ is usually the differences (errors) between the NN predictions and training data. Let $\mathcal{L} = \mathcal{L}(t, \mathbf{p})$ denotes the loss function, where t is the input time series and $\mathbf{p}$ is the parameters vectors containing in formations of $\mathcal{I}$, $\mathcal{D}$, and NN. If no external constraints or bounds are enforced, the optimization problem hence taking the form:

$$\min_{t \subset \mathbb{R}^{d_{out}}} \mathcal{L}(t, \mathbf{p}) \tag{5}$$

Minimizing $\mathcal{L}$ requires recursive iterations over the NN as in Equation (4): the information encoded in $\mathcal{L}$ reduces during iterations given the optimization methods; where we adopt limited-memory Broyden–Fletcher–Goldfarb–Shanno optimization algorithm, a quasi-Newton method (L-BFGS-B in `TensorFlow 1.x`) [57, 58]. The optimization are carried over iterations loop from the blue box (NN) to purple box ($\mathcal{I}$ & $\mathcal{D}$) to red box ($\mathcal{L}$) as in Figure 1. The maximum iterations are set as 2×10^5. In the PIDOC formulation, $\mathcal{L}$ is calculated based on mean square errors of the encoded information to be construed in Section 3.2.

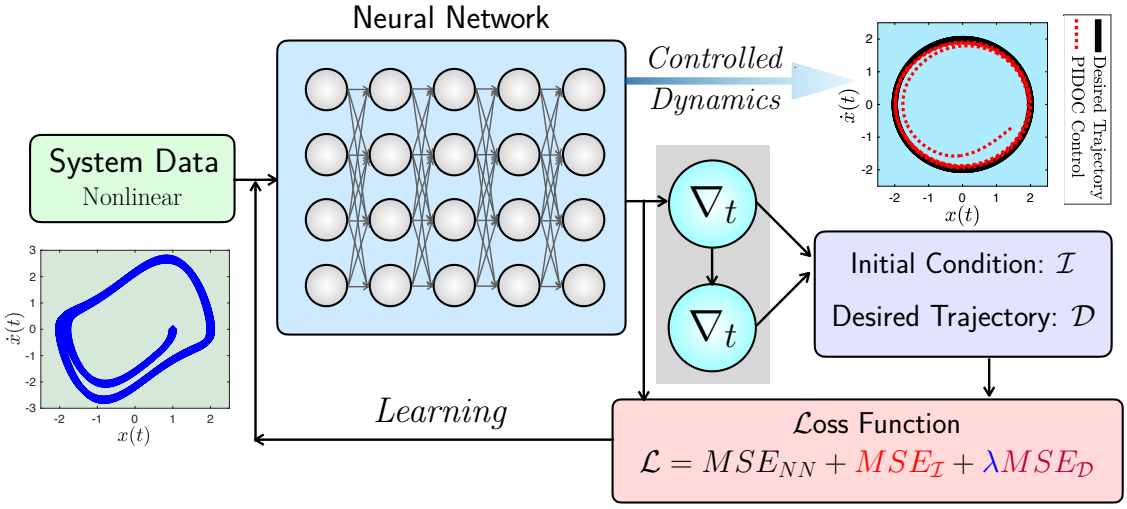

Figure 3: The schematic diagram for PHYSICS-INFORMED DEEP OPERATOR CONTROL framework was inspired by PINN. The system data of the nonlinear van der Pol oscillator is input to the deep learning framework, as automatic differentiation can encode physics information as in the purple box. The encoded information is further forward to the loss function, as the Langrangian multiplier λ can enlarge the control signal, which feedbacks to the deep learning scheme for learning the dynamics, which can output the ideal dynamics.

3.2 Physics-Informed control

The implementation of control signals are enabled through the physics information inserted in the loss function, mimicking the strategy of PINNs, but instead aim to execute command disparate

to the training data as to *tune* the system into a stable stage. The formulation of the loss function includes the mean square errors (MSE) of the NN generation MSE_{NN}, initial conditions $MSE_{\mathcal{I}}$ and the desired trajectory $MSE_{\mathcal{D}}$ multiplies a Lagrangian multiplier λ as to enlarge the control signal:

$$\mathcal{L} = MSE_{NN} + MSE_{\mathcal{I}} + \lambda MSE_{\mathcal{D}} \tag{6}$$

where the NN generation errors MSE_{NN} are computed as the MSE between the difference of training data x_{train} and NN predicted output x_{pred}:

$$MSE_{NN} := \frac{1}{N}\sum_{i=1}^{N} \left| x_{train} - x_{pred} \right|^2 \tag{7}$$

The initial position loss $MSE_{\mathcal{I}}$ are computed as the MSE between given initial conditions of $\mathcal{D}$ with system predictions x_{pred} at the initial:

$$MSE_{\mathcal{I}} := \frac{1}{N}\sum_{i=1}^{N} \left| x^0_{pred} - x^0_{\mathcal{D}} \right|^2 \tag{8}$$

where x^0_{pred} denotes the initial position (0^{th} in the array in `Python`) of NN prediction; $x^0_{\mathcal{D}}$ denotes the initial position of desired trajectory.

The control signal losses $MSE_{\mathcal{D}}$ are the MSE between the differences between the zeroth and second order derivatives of the position between the NN predictions (output) and desired control trajectories. In short, $MSE_{\mathcal{D}}$ imposes our desired control to PIDOC:

$$MSE_{\mathcal{D}} := \frac{1}{N}\sum_{i=1}^{N} \left| \left(\frac{dx^2_{\mathcal{D}}}{dt^2} - \frac{dx^2_{pred}}{dt^2} \right) + \left(x_{\mathcal{D}} - x_{pred} \right) \right|^2 \tag{9}$$

Here, $MSE_{\mathcal{D}}$ incorporates the control, and the multiplication of λ in Equation (6) allows us to tune the weight of the signal. The optimization problem formulated in Equation (5) can hence be considered as a multi-objective gradient-based optimization, as the objective of control can be tuned through λ. Specifically, it is reported that the limited memory BFGS method has been wide applied to large scale unconstrained optimizations [59].

Classical control approach impose a square wave or a triangular function as signals [60]. For circuits system specifically, applying a sinusoidal wave is a common practice, as also did by Cooper *et al.* [53]. Here, for applications of PIDOC, we also applied a sinusoidal wave, multiplies an adjustable multiplier Λ to control the amplitude of the phase:

$$x_{\mathcal{D}}(t) = \Lambda \sin(t), \implies \dot{x}_{\mathcal{D}}(t) = \Lambda \cos(t), \; \ddot{x}_{\mathcal{D}}(t) = -\Lambda \sin(t) \tag{10}$$

Given $x_{\mathcal{D}}$, the output phase portrait ($x(t)$ - $\dot{x}(t)$ diagram) is expected to be a circular trajectory. However, to make PIDOC adjustable with the desired trajectory amplitude Λ, one should modify Equation (7) as to make the NN losses contain information of trajectory amplitude, with the same training data:

$$MSE_{NN} := \frac{1}{N}\sum_{i=1}^{N} \left| x_{train} - \frac{x_{pred}}{\Lambda} \right|^2 \tag{11}$$

To perform decent system behavior analyses, we elicit four parameters for comparing behaviors involved in designing and applying PIDOC for control, as follows.

The accuracy of the controlled position (NN predictions) by PIDOC can be quantified by computing the absolute errors mean value between the desired position $x_{\mathcal{D}}$ with the PIDOC output x_{pred}, averaged over the data samples N:

$$|\overline{\mathcal{E}}| \equiv \left|\overline{\mathcal{E}}_{x(t)}\right| = \left|\frac{1}{N}\sum_{i=1}^{N}\left(\frac{x_{pred} - x_{\mathcal{D}}}{x_{\mathcal{D}}}\right)\right| \tag{12}$$

where $N = 3000$ in our case of applying PIDOC to van der Pol system.

The average value of the objective function (or loss function in the NN) $\mathcal{L}$ quantifies the accuracy of control (NN prediction) during the optimization process as in Equation (5), nominated as the mean losses $\overline{\mathcal{L}}$, calculated as the losses per iterations

$$\overline{\mathcal{L}} = \frac{1}{M}\sum_{i=1}^{M}(\mathcal{L}) \tag{13}$$

where M is the iterations number, which are not fixed and dependent on the convergent criteria of L-BFGS-B [57].

Computing power consumption can be quantified recalling the machine running time (`timeit.default_timer()` module in `Python`), which is symbolized as $\mathcal{T}$ as computation time. Note that the training of the NN is carried out on Google Colab [52], hence the unit of the exact time may not accurately reflect the computer platform, yet the quantity differences can qualitatively reflect the system behavior.

Since the computation time are collected over an uncertain iterations number M, averaging $\mathcal{T}$ over M, and normalized over a benchmark time $\hat{\mathcal{T}}^{\dagger}$ could proffer a decent quantification of computing time per iterations during the NN learning, called the mean normalized time $\tilde{\mathcal{T}}$:

$$\tilde{\mathcal{T}} = \frac{\mathcal{T}}{M}\frac{1}{\hat{\mathcal{T}}^{\dagger}} \tag{14}$$

where the computing time of the benchmark problem $\hat{\mathcal{T}}^{\dagger}$ is averaged through $\hat{\mathcal{T}}^{\dagger} = \mathcal{T}^{\dagger}/M^{\dagger}$, where $\mathcal{T}^{\dagger}$ is the total computing time and $M^{\dagger}$ is the iterations of the benchmark problem, to be elaborated in Section 3.3.

3.3 Numerical experiments

The systematic behavior of PIDOC is analyzed through numerical experiments covering two aspects: the capacity of PIDOC for controlling different systems, and how the hyperparameters and intrinsic structure of PIDOC variate its control process. Initially, we apply a benchmark problem for estimating the control process of PIDOC: the amplitude of the desired trajectory $\Lambda = 2$ in Equation (10); the van der Pol system with nonlinear parameter $\mu = 1$ in Equation (1); given an initial point of $(1, 0)$ corresponds to Figure 2; taking the Lagrangian multiplier $\lambda = 1$ as in Equation (6); with a neural network of 30 neurons per 6 layers (denoted as 6×30). We first apply PIDOC to control the benchmark problem and analyze its system behavior. Further, for how PIDOC behaves differently for different systems, we first change the trajectory amplitude Λ from 1 to 5 and study the PIDOC behavior; followed by changing initial positions as $(1, 0)$, $(5, 0)$ and $(0, 5)$ for studying how initial positions may affect PIDOC controls; followed by changing the nonlinearity on μ in Equation (1) to check how PIDOC controls different nonlinear systems. Regarding the PIDOC basic architecture and its influence on the control signals, we first change the

NN architecture (blue box in Figure 3) and apply PIDOC to the benchmark problem of $\mu = 1$; we further increase the layers as try to apply PIDOC for controlling system with high nonlinearities ($\mu = 5$ in Equation (1)). To study whether the enlargement of the control signal can increase or decrease the effectiveness of controls, the Lagrangian multiplier λ in Equation (6) is changed for five different values, $0, 1, 10, 10^3, \infty$, to test the PIDOC systematic behavior. Note that for $\lambda = \infty$, we only save the $MSE_{\mathcal{D}}$ term and eliminate the rests as this numerically represent $\lambda \longrightarrow \infty$. The estimation on van der Pol system and PIDOC architectures are based on the benchmark problem as only changing the parameters to be investigated. To eliminate the errors of reloading Google Colab for calculation of training time and other parameters, the numerical experiments are redone for each table to investigate each factor in each section.

4 Results and Discussion

4.1 System behavior analysis

The PIDOC framework is first applied to the benchmark problem as introduced in Section 3.3, the systematic behavior is tested based upon the phase portrait, the time scheme of position $x(t)$ and acceleration $\ddot{x}(t)$, respectively, both shown in Figure 4. From Figure 4 **A** in Figure 4 it can be detected that PIDOC successfully implement control to the desired trajectory as shown in the red dashed line converging to the circle of $\Lambda = 2$, getting rid of the chaotic behavior of the van der Pol system in blue dashed line. However, it can be noticed in Figure 4 **B** that the controlled scheme of PIDOC as in the red line exhibits a phase difference with the desired route as in the black line. It can also be observed in Figure 4 **C** the PIDOC controlled scheme displays an oscillating behavior as indicated in the red line.

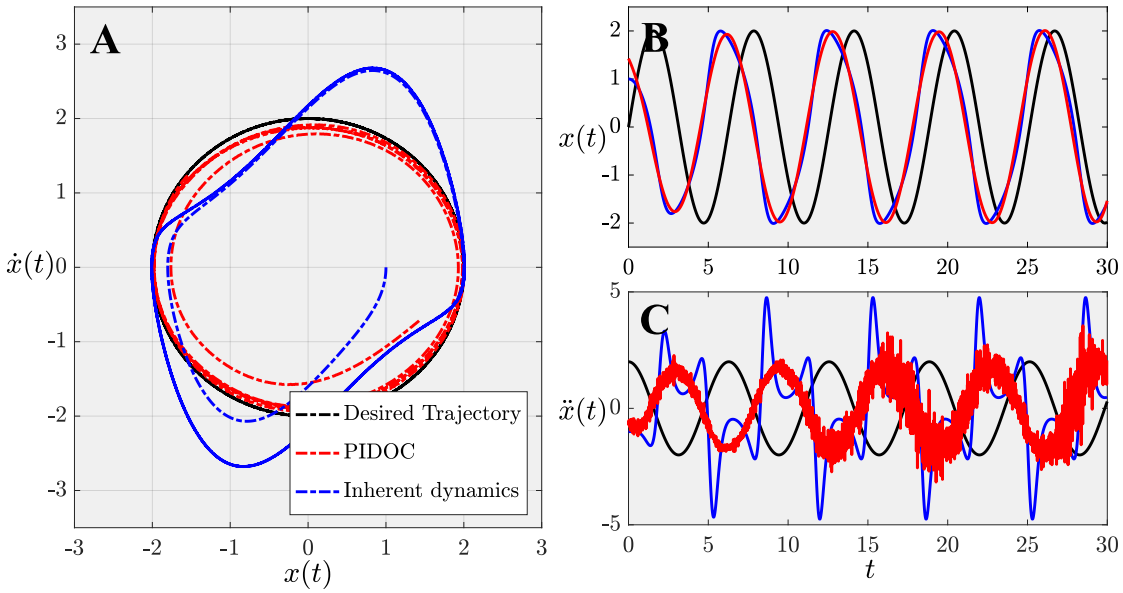

Figure 4: The system behavior of using PIDOC to control van der Pol dynamics applied to the benchmark problem. **A**. the phase portrait of the desired trajectory $\mathcal{D}$, PIDOC controlled signal, and the van der Pol inherent dynamics. **B**. the plot of position $x(t)$ regarding time t. **C**. the plot of acceleration $\ddot{x}(t)$ regarding time t.

Three characteristics of PIDOC control can be concluded from Figure 4: (1) PIDOC successfully

Λ	$\lvert\overline{\mathcal{E}}\rvert$	$\mathcal{T}$	$\overline{\mathcal{L}}$	$\tilde{\mathcal{T}}$
1	2.2501×10^{-4}	1.9554×10^{4}	2.3630×10^{3}	1.0000
$2^{\dagger}$	2.2520×10^{-4}	1.4098×10^{4}	1.0562×10^{3}	0.5642
3	2.2413×10^{-4}	2.0273×10^{4}	1.1336×10^{3}	0.9275
4	2.2330×10^{-4}	2.0759×10^{4}	1.4812×10^{3}	0.8074
5	2.2241×10^{-4}	2.1210×10^{4}	1.6920×10^{3}	0.8443

Table 1: Parameters estimation of the trajectory amplitude Λ. †The benchmark setup used for parameter tuning.

implemented the control to guide the system behave based upon $\mathcal{D}$; (2) PIDOC controlled dynamics exhibits a phase lag with the desired signals; (3) the acceleration $\ddot{x}(t)$ exhibits a fluctuating behavior. For (2) and (3), the following explanations are provided: the phase difference in (2) (Figure 4 **B**) are attributed to PIDOC aims to follow the given initial positions $\mathcal{I}$ encoded in Equation (6). The stochasticity for (3) observed in Figure 4 **C** can be attributed to both the stochastic nature of NN training and numerical differential of position $x(t)$: the weights $\mathbf{w}_{\mathfrak{L}}$ in Equation (4) are randomized and renewed per iterations as to approximate the nonlinear data given for training, as it seems smooth for first order approximation of $x(t)$ in Figure 4 **B**, the higher order terms $\ddot{x}(t)$ will enlarge the stochasticity of the signal. Moreover, the formulation of PIDOC as adopted from PINNs [22, 23, 27], automatic differentiation also elicit errors as encoded in $\mathcal{L}$ (Equation (6)) [61]. Both can also be accounted as factors as to explain fluctuations in Figure 4 **C**.

4.2 Nonlinearity and trajectory convergence

4.2.1 Amplitude of control trajectory

By changing amplitudes of the desired trajectories, PIDOC is tested on the capacities of executing different control signals, as indicated in Section 3.3. With the given five trajectories of $\Lambda = 1 \sim 5$, the phase portraits are shown in Figure 10 **A**. The losses corresponding to each trajectory during the NN training are shown in Figure 10 **B**. Figure 10 **A** contends that as trajectory amplitudes Λ increases there is a more evident difference between the PIDOC controls and $\mathcal{D}$. However, PIDOC exhibits good signal implementation and gets rid of the inherent dynamics successfully for both amplitudes of desired trajectories. Figure 10 **B** indicates the losses reduced to the lowest value as compared with other Λ with the least iterations, as $\Lambda = 2$ & 3 losses are in the same numerical value, same as $\Lambda = 4$ & 5, during a value between 10^2 to 10^3.

The absolute mean errors (Equation (12)), training time, mean losses (Equation (13)), and the mean normalized time (Equation (14)), corresponding to Figure 5, shortened as "PIDOC estimates", are shown in Table 1, which unveils information that are not straightforward in Figure 5. The $\lvert\overline{\mathcal{E}}\rvert$ values indicates for targeted trajectories with higher amplitudes, the relative errors are lower. For a higher trajectories' values, the training time $\mathcal{T}$ are higher. The average losses $\overline{\mathcal{L}}$ show good agreement with Figure 5 except for $\Lambda = 1$: an evidently higher mean loss than other trajectories. One can explain such a phenomenon with the instability of weights and biases generation of NN training caused the loss explosion, which is common for NN. For the normalized mean time $\tilde{\mathcal{T}}$, an unanticipated phenomenon of shorter $\tilde{\mathcal{T}}$ with higher amplitudes of trajectories is reported.

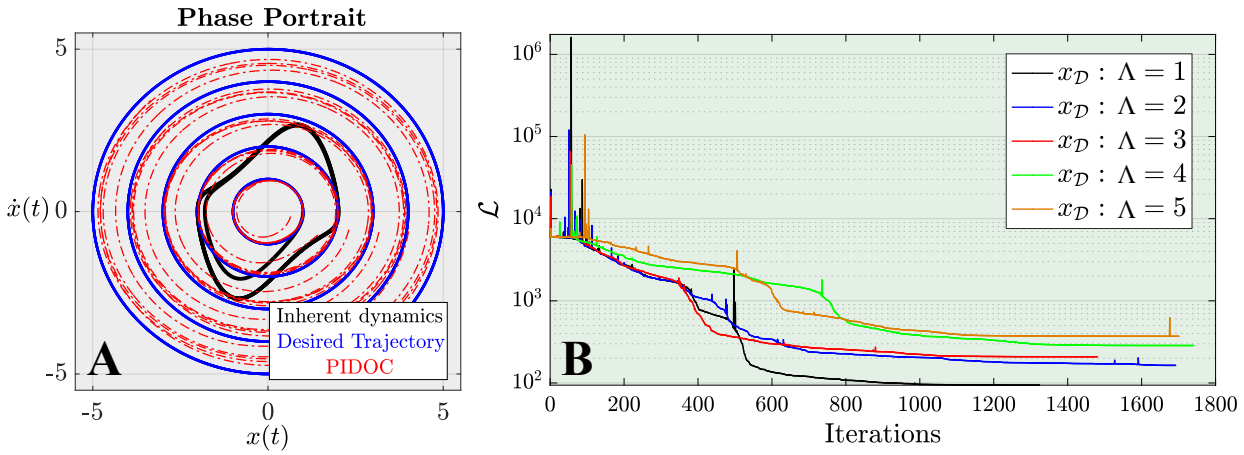

Figure 5: Systematic behavior analysis of PIDOC applied to different trajectory amplitudes Λ. **A**. the phase portrait of the inherent dynamics, five different desired trajectories $\Lambda = 1, 2, ..., 5$ marked in blue solid lines and the corresponding PIDOC controlled output marked in red dotted lines. **B**. the loss function - iterations diagram for five trajectories.

4.2.2 Initial positions

Given three different initial positions $\mathcal{I}$: $(1,0)$, $(5,0)$, $(0,5)$, the PIDOC control and systematic responses are shown in Figure 6. Note that the information of $\mathcal{I}$ are encoded to PIDOC within the physics-informed control in Equations (6), (8). Figure 6 **A** implies all PIDOC controls with different $\mathcal{I}$s successfully converge to the desired trajectory, with the control $\mathcal{I}$ of $(0,5)$ exhibits a slightly higher fluctuation as in the green line and the control $\mathcal{I}$ of $(5,0)$ exhibits a strong trajectory mismatch at the initial stage as indicated in the pink line. The losses in Figure 6 **B** implies when $\mathcal{I}$ is $(1,0)$ PIDOC exhibits the lowest loss while for $\mathcal{I}$ is $(5,0)$ the loss is the highest, which agrees well with Figure 6 **A**. Figure 6 **C** shows that PIDOC with $\mathcal{I}$ of $(5,0)$ and $(0,5)$ displays a slightly weaker fluctuations as in red and orange dashed lines compared with the green one. Specifically, both PIDOC controls for three $\mathcal{I}$s have an obvious phase difference, yet all converge to the desired trajectory. Recall our estimation for Figure 4, such a phase difference can be attribute to the approximation for initial positions by PIDOC as we encode such an information in Equation 8, which decently explains the phase difference in Figure 6 **C**.

$\mathcal{I}$	$\lvert\overline{\mathcal{E}}\rvert$	$\mathcal{T}$	$\overline{\mathcal{L}}$	$\tilde{\mathcal{T}}$
$(1,0)^{\dagger}$	2.2520×10^{-4}	1.4098×10^{4}	1.0562×10^{3}	1.0000
$(5,0)$	3.6666×10^{-4}	2.8535×10^{4}	7.1597×10^{3}	2.4050
$(0,5)$	4.0497×10^{4}	2.9442×10^{4}	2.5417×10^{3}	3.3887

Table 2: Parameters estimation of initial position $\mathcal{I}$. †The benchmark setup used for parameter tuning.

The PIDOC estimates corresponding to Figure 6 are shown in Table 2. It can be deduced from Table 2 that the for $\mathcal{I} = (0,5)$, PIDOC has the highest absolute mean error, with $\mathcal{I} = (1,0)$ has the lowest. The difference of the $\lvert\overline{\mathcal{E}}\rvert$ for points $(5,0)$ and $(0,5)$ may seem conflict with visualization in Figure 6 **A** and **B**. However, the PIDOC control for $\mathcal{I} = (5,0)$ (pink line in Figure 6 **A**) has a strong trajectory variation at the initial stage with good convergence to the desired trajectory later, which is also represented in the black dashed line in Figure 6 **C**, yet for $\mathcal{I} = (0,5)$ (green

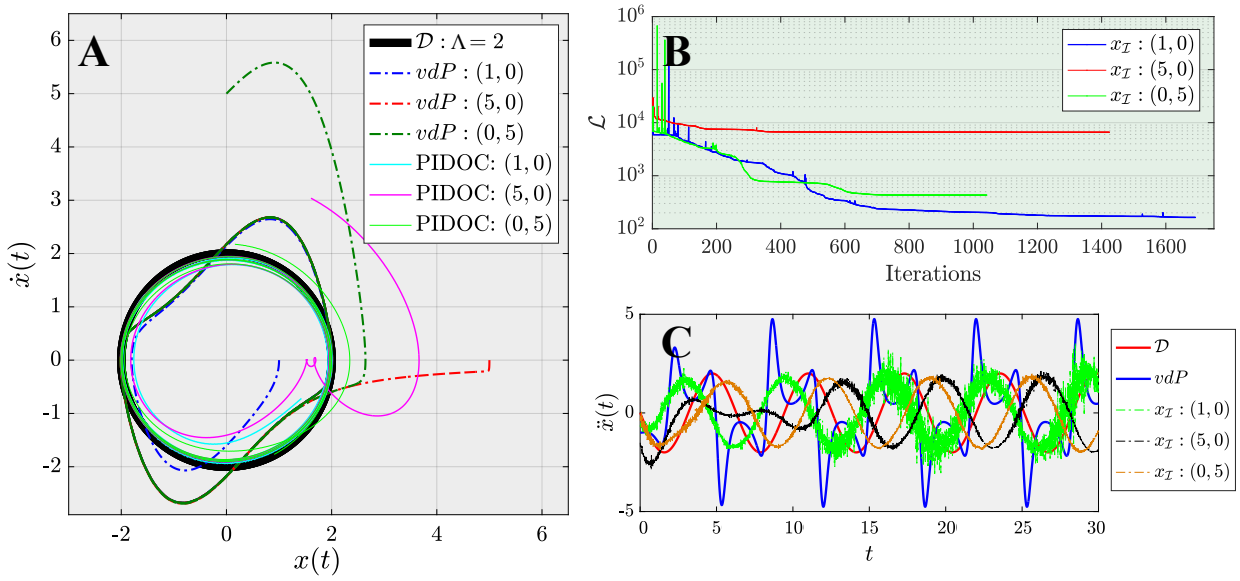

Figure 6: Systematic behavior estimation of PIDOC applied to the control signals $\Lambda = 2$ with different initial positions. Note that $\mathcal{D}$, vdP in plot legend stands for desired trajectory and van der Pol inherent dynamics. **A**. phase portrait of the desired trajectory, inherent dynamics given three different initial points $(1,0)$, $(5,0)$, $(0,5)$; and PIDOC controlled trajectories given three initial positions. **B**. the loss function - iterations diagram is given three different initial points. **C**. the acceleration $\ddot{x}(t)$ - time plot of desired trajectory $\mathcal{D}$, van der Pol inherent dynamics, and the three PIDOC applied controlled routes given different initial positions.

line in Figure 6 **A**) the trajectory fluctuation is evidently more obvious during the whole PIDOC controls, accounts for the higher $|\overline{\mathcal{E}}|$ value in Table 2. For $\mathcal{T}$ and $\tilde{\mathcal{T}}$ such a trend also stands. For $\overline{\mathcal{L}}$, $\mathcal{I} = (5,0)$ displays an evidently higher value, showing good agreement with Figure 6 **B**.

4.2.3 System nonlinearity

Nonlinearity is the key essence for control, especially for chaotic systems. For van der Pol oscillating circuits, systematic nonlinearities are represented with different μ values, varying from $\mu = 1, 3, 5, 7, 9$, for testing PIDOC controls. Figure 7 **A** visualized the phase portrait of such intrinsic nonlinear dynamics. After imposing the PIDOC controls, Figure 7 **B** shows the output control with different μs. Figure 7 **C** corresponds to Figure 7 **A**, showing how $x(t)$ evolution with time of the van der Pol inherent dynamics; with Figure 7 **D** corresponds to Figure 7 **D** showing the time evolution of $x(t)$ for PIDOC controls. Figure 7 **E** shows the loss functions - iterations diagrams of different PIDOC control for systems of different nonlinearities.

Figure 7 **E** shows an evident lower losses for system with low nonlinearity of $\mu = 1$, along with more iterations with $\mu = 1$ & 7. Comparing Figure 8 **A** and **C** we observe how high nonlinearities in phase portrait displayed on $x(t)$: a lower frequency with a specifc band structure of a "sharpe band shape" indicated in the waves in 8 **C** (specifically for the blue and orange dashed lines). From Figure 8 **B** we see a two "inner circles" attached on the two sides for high nonlinearities as can be observed in blue and orange dashed lines. Moving to Figure 8 **D** a specific "double wave" shape is observed, a smaller wave on a bigger wave as also be observed in blue and orange dashed lines. Comparing Figure 8 **B** and **D** one can conclude that the "double wave" structure observed

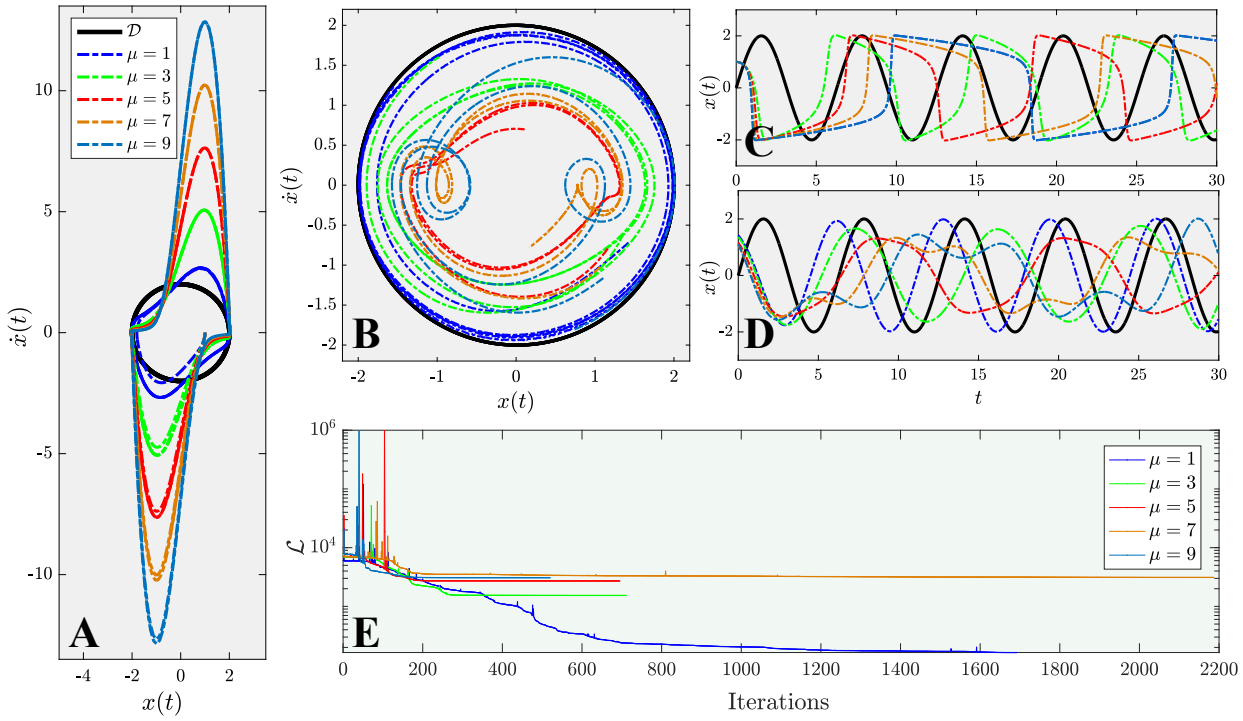

Figure 7: System analysis on PIDOC considering the nonlinearity of the van der Pol systems with different μ. **A**. the inherent dynamics of different van der Pol systems of different μ marked in dashed lines with different colors with the desired control trajectory $\mathcal{D}$ marked in black. **B**. the phase portrait of the desired trajectory corresponding to PIDOC applied to different systems of different nonlinearities. C. the position plot with time t of the system's inherent dynamics. **D**. the position plot with time t of the PIDOC controlled dynamics. **E**. the loss function - iterations plot of PIDOC applied to van der Pol systems of different nonlinearities. Note that the colors used for different PIDOC controls of different nonlinearities all correspond to subfigure **A**.

contributes to the smaller circles in the phase portrait. Comparing Figure 8 **D** and **C** one deduce that the reason of the "small wave" generation is the sharp band structure observed in Figure 8 **C**: the high nonlinearitity generates the sharp wave structure, make imposing the control signal of sinusoidal function significantly difficult.

μ	$\lvert\overline{\mathcal{E}}\rvert$	$\mathcal{T}$	$\overline{\mathcal{L}}$	$\tilde{\mathcal{T}}$
$1^{\dagger}$	2.2520×10^{-4}	1.4098×10^{4}	0.0106×10^{5}	1.0000
3	2.4732×10^{-4}	2.3042×10^{4}	0.0266×10^{5}	0.3889
5	2.1821×10^{-4}	2.4628×10^{4}	0.0579×10^{5}	0.4253
7	2.4079×10^{-4}	2.7107×10^{4}	0.0353×10^{5}	0.1488
9	2.9083×10^{-4}	3.2397×10^{4}	2.6962×10^{5}	0.7477

Table 3: Parameters estimation of the nonlinearity on μ. †The benchmark setup used for parameter tuning.

Corresponding to Figure 7, Table 3 shows the PIDOC estimates for different nonlinearities. It is observed that both $\mu = 1, 3, ..., 9$, the $\lvert\overline{\mathcal{E}}\rvert$ values are in the same range, with a slight difference in numerical values, which is not expected intuitively based on Figure 7. The $\mathcal{T}$ values increase with nonlinearities. The mean losses $\overline{\mathcal{L}}$ variate but shows an increasing trend with higher nonlinearity, with a loss explosion observed for $\mu = 9$ as explained for Table 1. The mean normalized time $\tilde{\mathcal{T}}$ shows that the computational burden reduces per iteration with higher nonlinearities.

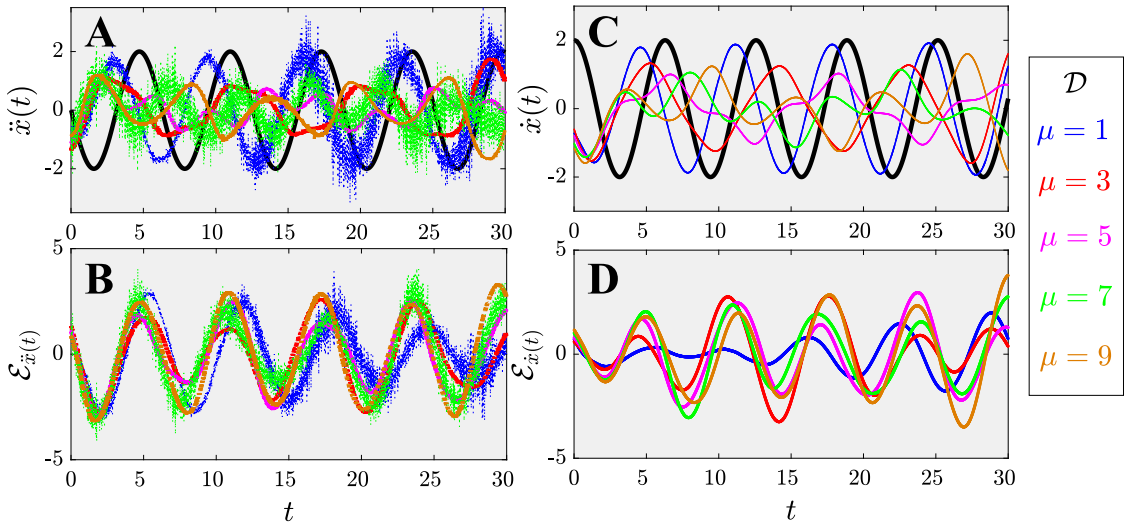

Figure 8: The plot of velocity $\dot{x}(t)$ and acceleration $\ddot{x}(t)$ to their corresponding errors regarding time t considering cases of different nonlinearities. Note that the colors for cases in van der Pol systems of different nonlinearities are shown in the legend on the right side, where $\mathcal{D}$ stands for the desired control trajectory. **A**. the acceleration $\ddot{x}(t)$ - t diagram. **B**. the errors of the acceleration. **C**. the acceleration $\dot{x}(t)$ - t diagram. **D**. the errors of the velocity.

To further explore the similar values of $\lvert\overline{\mathcal{E}}\rvert$, we create Figure 8. Figure 8 **A** and **C** shows the acceleration and velocities of PIDOC controls marked in dashed lines in different colors in the right legend compared with the desired trajectory marked in black solid lines. Figure 8 **B** and **D** shows the difference (or errors) for velocities and accelerations calculated by

$$\mathcal{E}_{\ddot{x}(t)} = \ddot{x}_{\mathcal{D}}(t) - \ddot{x}_{pred}(t), \quad \mathcal{E}_{\dot{x}(t)} = \dot{x}_{\mathcal{D}}(t) - \dot{x}_{pred}(t) \tag{15}$$

It is reported that the acceleration errors $\mathcal{E}_{\ddot{x}(t)}$ of different systems with different nonlinearities exhibits the same frequency with the control signal, comparing Figure 8 **A** and **B**. The errors of velocities $\mathcal{E}_{\dot{x}(t)}$ does not show such evident trends yet all seemingly fluctuating in the same frequency as in Figure 8 **D**. We can therefore conclude a significant characteristic of PIDOC controls: due to the imposition of $\mathcal{I}$, there will be a phase lag for PIDOC controls, which is reported in Figure 4. Such phase lags generate a so-called error, or difference, between the control signal and PIDOC control. For van der Pol systems of different nonlinearities, the errors exhibit the same frequency, with similar wave range values. The similarities of the range values and frequency combined explain why PIDOC controls exhibit similar errors as reported in Table 3. To note, such a phase lag successfully implements the controls for relative low nonlinearities as in Figures 4, yet still exhibits imperfect controls for high nonlinearities.

4.3 Hyperparameters and control

4.3.1 Deep neural networks

To test how the NN structures variate the control process, two cases are set up for investigation: (1) the benchmark problem with reduced neurons and layers, of six different sets: NN structures of 1×30, 3×30, 6×30, 1×10, 3×10, 6×10; and (2) the increased neurons and layers for controlling van der Pol system with high nonlinearity of $\mu = 5$, of five different sets: NN layers of 6, 9, 15, 30, 50, with 30 neurons per layers. The aim of case (1) is to investigate whether reduced neurons and layers in NN, as one expects reduced capabilities of approximating nonlinear data, can still implement controls of high quality, as observed in Figure 4. Particularly, it has been reported by Pinkus [62] that a single hidden layer NN can approximate nonlinear mappings, followed by a physics-informed practice by Lu *et al.* [27]. We therefore specifically test PIDOC with a single hidden layer for testing its ability for controlling the van der Pol system. The aim of the case (2) is to investigate whether increased approximation capacity can tackle the control of highly nonlinear systems as we made effort in Figure 7.

Figure 9 **A** and **B** shows for reduced NN layers and neurons all exhibits good control implementations, with a slightly trajectory variation at the beginning stage for NN structure 3×10 as indicated in the pink line with higher losses. Figure 9 **C** plot the intrinsic dynamics of van der Pol system with $\mu = 5$, as comparing with the controlled phase in 9 **D**: with increasing layers the controlled phase shows reduced nonlinearities, especially for layers $L = 6$ and 50 for comparing the light and pure blue dashed lines - the vortex-liked shape on the two sides of the phase when $x \approx 1.2$ evidently reduced with increasing layers. For different layers the losses show a similar trends as reported in Figure 9 **E**. Notably, Figure 9 **A** also indicate that single hidden layer NN shows good approxibilities, with better phase control than NN structure of 3×10.

Layers	Neurons	$\lvert\overline{\mathcal{E}}\rvert$	$\mathcal{T}$	$\overline{\mathcal{L}}$	$\tilde{\mathcal{T}}$
1	30	$2.2472 \times 10{-}4$	1.2558×10^4	691.4871	0.9229
3	30	$2.2474 \times 10{-}4$	1.2853×10^4	3.8015×10^5	1.2450
$6^\dagger$	$30^\dagger$	$2.2520 \times 10{-}4$	1.1098×10^4	1.0562×10^3	1.0000
1	10	$2.2441 \times 10{-}4$	1.0823×10^4	1.1010×10^3	0.7572
3	10	$2.1779 \times 10{-}4$	1.1285×10^4	1.9470×10^3	1.0958
6	10	$2.2439 \times 10{-}4$	1.1444×10^4	6.6087×10^3	1.0144

Table 4: Parameters estimation of the NN structure considering layers and neurons to the benchmark setup. †The benchmark setup used for parameter tuning where $\mu = 1$.

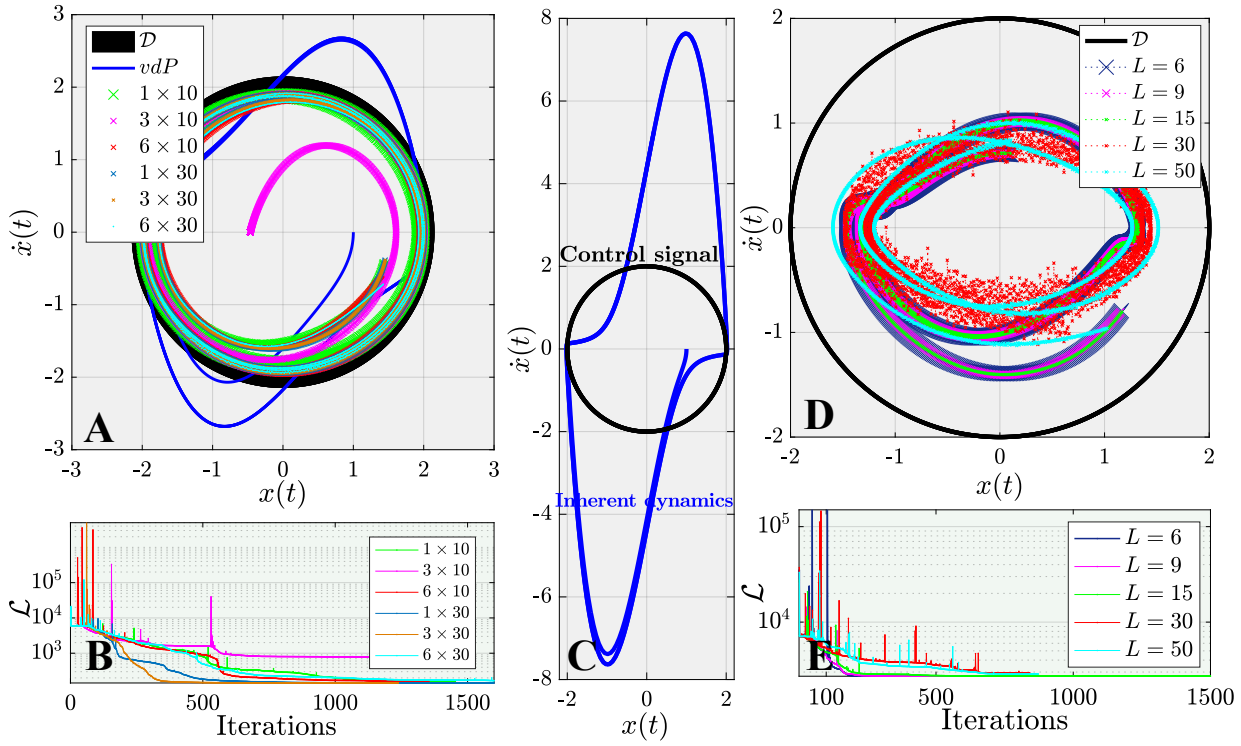

Figure 9: System behavior analysis of how the NN structure tune the control process of PIDOC for $\mu = 1$ and $\mu = 5$. **A**. the phase portrait of PIDOC controls considering different NN structures; the solid black line $\mathcal{D}$ denoted the desired trajectory, the blue solid line vdP denotes the van der Pol inherent dynamics. Different colored cross dots as marked in the legend denotes different NN structures. Note that subfigure **A** and **B** share the same color representation of the NN structure. **B**. the loss function - iterations diagram for different NN structures as applied to the benchmark problem of $\mu = 1$. **C**. the inherent van der Pol dynamics (marked as a blue solid line) when $\mu = 5$. **D**. the phase portrait of PIDOC applied to highly nonlinear van der Pol system of $\mu = 5$ corresponds to subfigure **C**. Note that the black solid line $\mathcal{D}$ is the desired trajectory. Different NN structures are denoted by cross dots in different colors as indicated in the legend. **E**. the loss function - iterations diagram corresponds to subfigure **D**.

Numerical investigation of Figure 9 **A** and **B** represented by PIDOC estimates for $\mu = 1$ are shown in Table 4. The $|\overline{\mathcal{E}}|$ and $\mathcal{T}$ are in approximately the same range for different NN structures. The losses evidently higher for NN of 3×30 and 6×10. The higher losses for NN of 3×10 can be captured in Figure 9 **B**; yet for bigger $\mathcal{L}$ for NN of 3×30 and 6×10, we can account it for the high loss fluctutations at the beginning stage, as also reported Table 1. For $\tilde{\mathcal{T}}$ we reports NN structure 3×30 took higher training time per iterations, and for 3×10 and 6×10 the $\tilde{\mathcal{T}}$ values are slightly higher.

Layers	Neurons	$\lvert\overline{\mathcal{E}}\rvert$	$\mathcal{T}$	$\overline{\mathcal{L}}$	$\tilde{\mathcal{T}}$
$6^{\dagger}$	$30^{\dagger}$	2.3083×10^{-4}	1.9409×10^{4}	3.1732×10^{3}	1.0000
9	30	2.3091×10^{-4}	1.9591×10^{4}	8.6195×10^{3}	6.9859
15	30	2.3055×10^{-4}	2.0028×10^{4}	3.4388×10^{3}	5.4510
30	30	2.2702×10^{-4}	2.0759×10^{4}	8.2714×10^{3}	3.4597
50	30	2.2431×10^{-4}	2.1228×10^{4}	3.9537×10^{3}	33.7096

Table 5: Parameters estimation of the NN structure regarding layers and neurons, estimating a highly nonlinear van der Pol system with $\mu = 5$. †The benchmark NN structure used for parameter tuning, adopting the van der Pol system of high nonlinearities.

The PIDOC estimates of highly nonlinear van der Pol system of $\mu = 5$ are shown in Table 5. The values $|\overline{\mathcal{E}}|$ are basically in the same range. A generally higher training time $\mathcal{T}$ and normalized training time per iterations $\tilde{\mathcal{T}}$ are reported for increasing layers. The increase on $\tilde{\mathcal{T}}$ indicates the optimization described in Equation 5 stops earlier with more layers, resulting in fewer iterations.

4.3.2 Lagrangian multiplier

It is intuitive to think that by enlarging the control signal we might expect a better control implementation. Curious about the effects of control signals, we applied different Lagrangian multiplier $\lambda = 0, 1, 10, 10^3, \infty$, for testing the systematic control accuracy. For $\lambda = 0$, there are simply no control signals encoded in the loss, where PIDOC is reduced to a normal NN with only the initial position encoded as a soft constraint. The problem thence turned into a standard NN learning and fitting problem. For $\lambda = \infty$, we simply simply eliminate Equations (8) and (9), as the control signals turn into infinity. The phase portrait, the time evolution of position $x(t)$ and $\ddot{x}(t)$ of different λs are shown in Figure 10.

Figure 10 **A** shows the phase portrait of PIDOC controls with different λs, indicates the pure NN learning of van der Pol dynamics displays display a fluctuation at the area around $[x(t), \dot{x}(t)] \approx [0.5, -1.5]$ for the red dashed line. The pink and green lines indicate for $\lambda = 1$ & 10 PIDOC implements generally good controls as both converge to the circular trajectories. However, as shown in the zoomed view in Figure 10 **B** one can discern with very high λ the PIDOC is difunctionalized as the controlled trajectory shrink to a very low value ($\approx 10^{-4}$). Figure 10 **C** and **D** are the positions $x(t)$ and accelerations $\ddot{x}(t)$ for different weights of control signals (λ). Both the subfigures indicate when λ approach a high value (10^3 & ∞) the positions and accelerations shrink to a very low value as can be observed from the blue and orange lines, corresponds to Figure 10 **A**. Notably, it has also been observed that a robust acceleration fluctuation occurs at $t \approx 21$ in the red dashed line, corresponds to the phase fluctuations in Figure 10 **A**, as the errors of NN approximations. Such errors can also be attributed to the stochastic nature of NN as we previously explained for Figure 4.

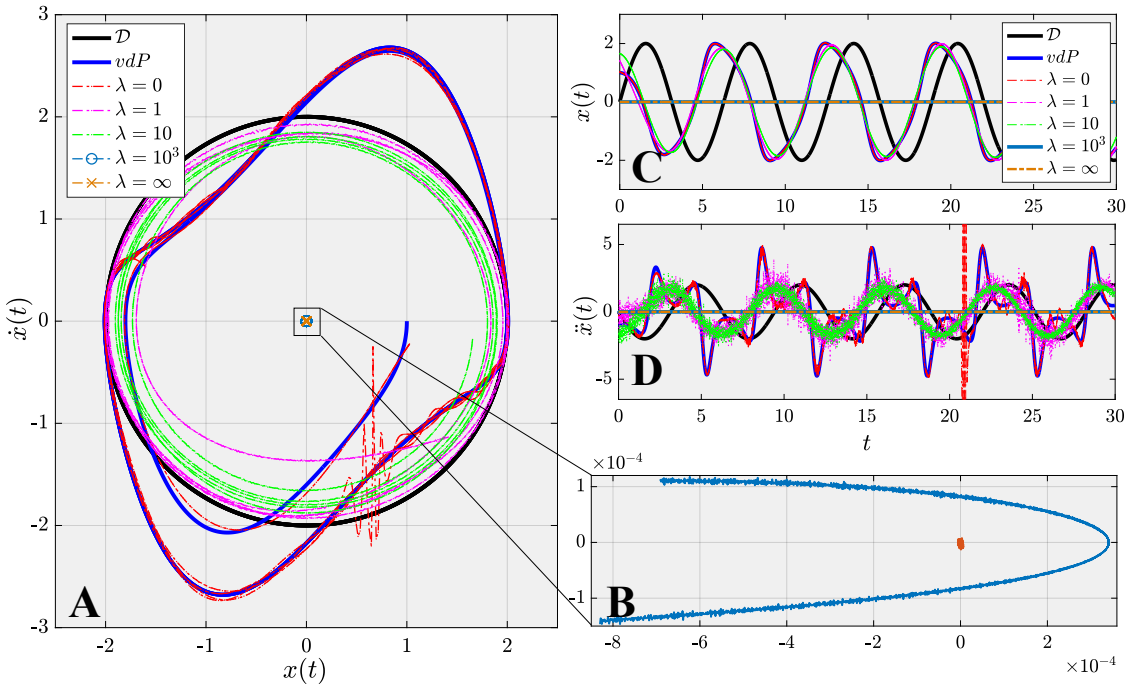

Figure 10: System behavior analysis of the effects of the Lagrangian multiplier on the control signal implementation for PIDOC. Note that the black solid line $\mathcal{D}$ is the desired trajectory, the blue solid line vdP denotes the van der Pol inherent dynamics, and the controlled route of different Lagrangian multiplier values are denoted in different colored dashed lines. Note that all the colors in the subfigures are marked the same as represented in the legend in subfigure **A**. **A**. the phase portrait of the PIDOC controls. **B**. zoomed view for rescale for Lagrangian multiplier $\lambda = 10^3$ and $\lambda = \infty$. **C**. the position $x(t)$ regarding time t for desired trajectory $\mathcal{D}$, van der Pol inherent dynamics and different PIDOC controls. **D**. the acceleration $\ddot{x}(t)$ regarding time t for desired trajectory $\mathcal{D}$, van der Pol inherent dynamics and different PIDOC controls.

λ	$\lvert\overline{\mathcal{E}}\rvert$	$\mathcal{T}$	$\overline{\mathcal{L}}$	$\tilde{\mathcal{T}}$
0	1.0181×10^{-4}	7.0409×10^3	415.1019	0.6867
$1^\dagger$	1.0221×10^{-4}	6.2442×10^3	5835.0830	1.0000
10	0.8626×10^{-4}	6.4437×10^3	2419.3699	0.9709
10^3	2.1127×10^{-4}	6.5846×10^3	45516.0465	47.2085
∞	2.1127×10^{-4}	6.7030×10^3	3.7753	52.9858

Table 6: Parameters estimation of the Lagrangian multiplier λ or enlarging or eliminating the effects of the control signal in the physics-informed loss. †The benchmark setup used for parameter tuning.

Table 6 numerically unveils how the weights of control signals affect PIDOC estimates. From the values of $|\overline{\mathcal{E}}|$ one can deduce for $\lambda = 10^3$ and ∞ there are an evidently higher errors. The training time $\mathcal{T}$ basically holds the same for both cases. However, it should be noted that there is an increasing $\overline{\mathcal{L}}$ as λ increases, yet when the control signal is eliminated, $\overline{\mathcal{L}}$ reduced to a very low value, as the problem turned into pure NN learning. The $\tilde{\mathcal{T}}$ values increases significantly for $\lambda = 10^3$ and ∞, which connected with Figure 10 **A** and **B** indicating increasing computational burden per iterations and low quality control caused by the high weights of control signals in PIDOC. From such results we can further propose an explanation for implementations of physics-informed controls: the high weights of control signals lead to the deprivation of information of the training data. Such deprivation may "confuse" the learning of NN as it is mainly designed for stochastic data-based learning and shows robust capabilities given a humongous dataset given no external constraint [20, 23]. Hence, as the core of deep learning, even the goal emphasizes control, the given data is always of key essence. We hence conclude that even the given training data of the van der Pol system is nonlinear it still contributes largely to the successful implementation of PIDOC controls.

5 Concluding Remarks and Future Works

In this paper we tackle a century-old yet widely applied problem, controlling a nonlinear van der Pol dynamical system, with a novel approach using Physics-Informed Neural Networks. Instead of adopting the traditional paradigm of learning and predicting using PINN, we use PINN for controlling nonlinear systems. A new PINN-based framework PHYSICS-INFORMED DEEP OPERATOR CONTROL, shortened as PIDOC, is presented, consisting of a deep neural network and the physics-informed control, including the desired control trajectories and initial positions. PIDOC is fed with systematic nonlinear data to control van der Pol circuits to output the controlled signals. To investigate the behavior and properties of PIDOC, we first applied PIDOC for benchmark control problems for systematic analysis, then designed three sets of numerical experiments for testing the effects of amplitudes of desired trajectories Λ, different initial points $\mathcal{I}$, and system nonlinearities as represented by μ. We then tune the hyperparameters to change the neurons and layers of the NN to study two problems: (1) does a NN with smaller volume still shows the same capability of controlling chaos applied to the benchmark problem; (2) can increase NN volume demonstrate better capabilities on controlling van der Pol systems with high nonlinearities. We also intend to verify the capability of single hidden layer NN to approximate nonlinear systems for part of the control. We also change Lagrangian multiplier λ as a weight factor to check how desired trajectories as control signals guide PIDOC in the control process.

Results indicate PIDOC controls exhibit higher stochasticity for higher-order terms, as can be attributed to the stochastic nature of deep learning, with a successful implementation of the desired trajectory on the benchmark problem. PIDOC also demonstrates capacity on increased trajectory amplitudes with lower absolute mean errors. For systems with different initial points, numerical experiments show for points further away PIDOC can still successfully implement controls as higher fluctuations at the initial stage. However, as we increase the system nonlinearities, the PIDOC output controls are not as ideal as the benchmark problems, as two vortex-liked structures occur on the phase portrait, with an evidently higher loss for systems with high nonlinearities. A decreased NN in PIDOC volume also shows good control implementations with the van der Pol system of $\mu = 1$, while increasing the layers did not make systems of high nonlinearity with $\mu = 5$ follow the desired trajectory as well as the benchmark problem. It should be noted that increasing layers do generate improvement on the output controlled signals as the vortex-liked structures in

phase portrait vanished, making the system more predictive. Increasing the weights of the control signals in PIDOC does not improve the control qualities based on the output. A further conclusion is made, even the systematic data is nonlinear and chaotic, it still contributes much to the PIDOC controls as PIDOC is intrinsically a deep learning-based control method.

Considering the successful implementation of PIDOC to control different van der Pol systems, further investigation on using PIDOC to impose control to other systems such as the Lorentz system would have brought in more insights into PIDOC. Also, comparison on the control properties based on PIDOC and deterministic controls, i.e., Cooper *et al.* [53] could also be potential researches. Improvement on PIDOC or further developed models tackling systems with high nonlinearities is of significance for future research.

Data Availability

All the code and data will be made publicly available upon acceptance of the manuscript through https://github.com/hanfengzhai/PIDOC. For the original PINNs code please refer to https://github.com/maziarraissi/PINNs. The training of the deep learning algorithms and simulation of the van der Pol system is conducted on Google Colab [52].

References

[1] Williams-Stuber, K., & Gharib, M. (1990). Transition from order to chaos in the wake of an airfoil. Journal of Fluid Mechanics, 213, 29-57. doi:10.1017/S0022112090002208.

[2] F. Argoul, A. Arneodo, P. Richetti, J. C. Roux, and Harry L. Swinney. Chemical chaos: from hints to confirmation. Acc. Chem. Res. 1987, 20, 12, 436–442. https://doi.org/10.1021/ar00144a002

[3] Merali, Z. Robotic roach creates order from chaos. Nature (2010). https://doi.org/10.1038/news.2010.15

[4] Toker, D., Sommer, F.T. & D'Esposito, M. A simple method for detecting chaos in nature. Commun Biol 3, 11 (2020). https://doi.org/10.1038/s42003-019-0715-9

[5] Barbara Błaz˙ejczyk, Tomasz Kapitaniak, Jerzy Wojewoda, John Brindley. Controlling Chaos in Mechanical Systems. Appl. Mech. Rev. Jul 1993, 46(7): 385-391. https://doi.org/10.1115/1.3120367

[6] Matsumoto, T.. "Chaos in electronic circuits." Proceedings of the IEEE 75 (1987): 1033-1057.

[7] W. A. Lin and J. B. Delos. Order and chaos in semiconductor microstructures. Chaos 3, 655 (1993); https://doi.org/10.1063/1.165994

[8] J. Mork, B. Tromborg and J. Mark, "Chaos in semiconductor lasers with optical feedback: theory and experiment," in IEEE Journal of Quantum Electronics, vol. 28, no. 1, pp. 93-108, Jan. 1992, doi: 10.1109/3.119502.

[9] Chaos. Encyclopedia of Mathematics. URL: http://encyclopediaofmath.org/index.php?title=Chaos&oldid=46308

[10] González-Miranda, J.M. (2004). Synchronization and Control of Chaos: An Introduction for Scientists and Engineers. London: Imperial College Press.

[11] Lorenz, Edward N. (1963). "Deterministic non-periodic flow". Journal of the Atmospheric Sciences. 20 (2): 130–141. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

[12] Ivancevic, Vladimir G.; Tijana T. Ivancevic (2008). Complex nonlinearity: chaos, phase transitions, topology change, and path integrals. Springer. ISBN 978-3-540-79356-4.

[13] Safonov, Leonid A.; Tomer, Elad; Strygin, Vadim V.; Ashkenazy, Yosef; Havlin, Shlomo (2002). "Multifractal chaotic attractors in a system of delay-differential equations modeling road traffic". Chaos: An Interdisciplinary Journal of Nonlinear Science. 12 (4): 1006–1014.

[14] Tufan Dogruer, Nusret Tan. Design of PI Controller using Optimization Method in Fractional Order Control Systems Author links open overlay panel. IFAC-PapersOnLine. Volume 51, Issue 4, 2018, Pages 841-846. https://doi.org/10.1016/j.ifacol.2018.06.124

[15] Araki, M. "PID Control". http://www.eolss.net/ebooks/Sample%20Chapters/C18/E6-43-03-03.pdf

[16] Bennett, Stuart (1996). "A brief history of automatic control". IEEE Control Systems Magazine. 16 (3): 17–25. doi:10.1109/37.506394

[17] Slotine, J. Applied Nonlinear Control; Prantice-Hall: Englewood Cliffs, NJ, USA, 1991; Chapter 9.

[18] Mall, K, Grant, MJ, Taheri, E. Solving complex optimal control problems with nonlinear controls using trigonometric functions. Optim Control Appl Meth. 2021; 42: 616– 628. https://doi.org/10.1002/oca.2692

[19] Su Whan Sung and In-Beum Lee. Limitations and Countermeasures of PID Controllers. Ind. Eng. Chem. Res. 1996, 35, 8, 2596–2610. https://doi.org/10.1021/ie960090+

[20] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539

[21] M. Hassaan and I. Elghandour, "A Real-Time Big Data Analysis Framework on a CPU/GPU Heterogeneous Cluster: A Meteorological Application Case Study," 2016 IEEE/ACM 3rd International Conference on Big Data Computing Applications and Technologies (BDCAT), 2016, pp. 168-177.

[22] 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.

[23] Karniadakis, G.E., Kevrekidis, I.G., Lu, L., et al. Physics-informed machine learning. Nat. Rev. Phys. **3**, 422–440 (2021).

[24] Chen, Z., Liu, Y. & Sun, H. Physics-informed learning of governing equations from scarce data. Nat Commun 12, 6136 (2021). https://doi.org/10.1038/s41467-021-26434-1

[25] Raissi, M., Yazdani, A., & Karniadakis, G. E. (2020). Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. Science (New York, N.Y.), 367(6481), 1026–1030. https://doi.org/10.1126/science.aaw4741

[26] Wang, R., Kashinath, K., Mustafa, M., et al. (2020) Towards Physics-informed Deep Learning for Turbulent Flow Prediction. KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. August 2020. pp 1457–1466.

[27] Lu Lu, Xuhui Meng, Zhiping Mao, and George Em Karniadakis. DeepXDE: A Deep Learning Library for Solving Differential Equations. SIAM Review 2021 63:1, 208-228.

[28] Laurent Pagnier, Michael Chertkov. Physics-Informed Graphical Neural Network for Parameter & State Estimations in Power Systems. arXiv preprint. arXiv:2102.06349

[29] Lee, J. Physics-Informed Neural Network for High Frequency Noise Performance in Quasi-Ballistic MOSFETs. Electronics 2021, 10, 2219. https://doi.org/10.3390/electronics10182219

[30] Zhai, H., Zhou, Q., Hu, G. BubbleNet: Inferring micro-bubble dynamics with semi-physics-informed deep learning. arXiv preprint. arXiv:2105.07179

[31] Anshul Choudhary, John F. Lindner, Elliott G. Holliday, Scott T. Miller, Sudeshna Sinha, and William L. Ditto. Physics-enhanced neural networks learn order and chaos. Phys. Rev. E 101, 062207. https://doi.org/10.1103/PhysRevE.101.062207

[32] Scott T. Miller, John F. Lindner, Anshul Choudhary, Sudeshna Sinha, William L. Ditto, The scaling of physics-informed machine learning with data and dimensions, Chaos, Solitons & Fractals: X, Volume 5, 2020, 100046, ISSN 2590-0544, https://doi.org/10.1016/j.csfx.2020.100046.

[33] Maziar Raissi, Paris Perdikaris, George Em Karniadakis. Multistep Neural Networks for Data-driven Discovery of Nonlinear Dynamical Systems. arXiv preprint. arXiv:1801.01236

[34] Fangzheng Sun, Yang Liu, Hao Sun. Physics-informed Spline Learning for Nonlinear Dynamics Discovery. Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21). https://www.ijcai.org/proceedings/2021/0283.pdf

[35] Eric Aislan Antonelo, Eduardo Camponogara, Laio Oriel Seman, Eduardo Rehbein de Souza, Jean P. Jordanou, Jomi F. Hubner. Physics-Informed Neural Nets for Control of Dynamical Systems. arXiv preprint. arXiv:2104.02556.

[36] M. T. Hagan and H. B. Demuth, "Neural networks for control," Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), 1999, pp. 1642-1656 vol.3, doi: 10.1109/ACC.1999.786109.

[37] D. H. Nguyen and B. Widrow, "Neural networks for self-learning control systems," in IEEE Control Systems Magazine, vol. 10, no. 3, pp. 18-23, April 1990, doi: 10.1109/37.55119.

[38] P.J. Antsaklis, "Neural Networks in Control Systems," Guest Editor's Introduction, I EEE C ontrol S ystems M agazine , Vol.10, No.3, pp.3-5, April 1990; Special Issue on 'Neural Networks in Control Systems' of the IEEE Control Systems Magazine , Vol.10, No.3, pp.3-87, April 1990.

[39] van der Pol, B. On "Relaxation Oscillations" I. Philos. Mag. 1926, 2, 978–992.

[40] van der Pol, B. The nonlinear theory of electric oscillations. Proc. IRE 1934, 22, 1051–1086.

[41] van der Pol, B.; van der Mark, J. Frequency de-multiplication. Nature 1927, 120, 363–364.

[42] Novak, Tyson (2008) Design principles of biochemical oscillators. Nature review 9:981-991.

[43] Kevin Rompala, Richard Rand, Howard Howland. Dynamics of three coupled van der Pol oscillators with application to circadian rhythms. Communications in Nonlinear Science and Numerical Simulation 12 (2007) 794–803. https://doi.org/10.1016/j.cnsns.2005.08.002.

[44] B.Z. Kaplan, I. Gabay, G. Sarafian, D.Sarafian. Biological applications of the "Filtered" Van der Pol oscillator. Journal of the Franklin Institute. Volume 345, Issue 3, May 2008, Pages 226-232. https://doi.org/10.1016/j.jfranklin.2007.08.005

[45] T.P. Chagas, B.A. Toledo, E.L. Rempel, A.C.-L. Chian, J.A. Valdivia. Optimal feedback control of the forced van der Pol system. Chaos, Solitons & Fractals. Volume 45, Issues 9–10, September–October 2012, Pages 1147-1156. https://doi.org/10.1016/j.chaos.2012.06.004

[46] M. Sayed, S. K. Elagan, M. Higazy and M. S. Abd Elgafoor. Feedback Control and Stability of the Van der Pol Equation Subjected to External and Parametric Excitation Forces. International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 6 (2018) pp. 3772-3783.

[47] A. Batool, A. Hanif, M. T. Hamayun and S. M. N. Ali, "Control design for the compensation of limit cycles in Van Der Pol oscillator," 2017 13th International Conference on Emerging Technologies (ICET), 2017, pp. 1-6, doi: 10.1109/ICET.2017.8281717.

[48] Duke University. The van der Pol System. URL: https://services.math.duke.edu/education/ccp/materials/diffeq/vander/vand1.html

[49] Schult D. Math 329 – Numerical Analysis webpage. URL: http://math.colgate.edu/math329/exampleode.py

[50] Mario Mulansky and Karsten Ahnert (2014), Odeint library. Scholarpedia, 9(12):32342.

[51] Daniel Müller-Komorowska. Differential Equations with SciPy – odeint or solve_ivp. Scientific Programming Blog. URL: https://danielmuellerkomorowska.com/2021/02/16/differential-equations-with-scipy-odeint-or-solve_ivp/

[52] Bisong E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_7

[53] Cooper, M.; Heidlauf, P.; Sands, T. Controlling Chaos—Forced van der Pol Equation. Mathematics 2017, 5, 70. https://doi.org/10.3390/math5040070

[54] Heidlauf, P.; Cooper, M. Nonlinear Lyapunov Control Improved by an Extended Least Squares Adaptive Feed forward Controller and Enhanced Luenberger Observer. In Proceedings of the International Conference and Exhibition on Mechanical & Aerospace Engineering, Las Vegas, NV, USA, 2–4 October 2017.

[55] Sands, T. Physics-Based Control Methods. In Advances in Spacecraft Systems and Orbit Determination; Rushi, G., Ed.; In-Tech Publishers: Rijeka, Croatia, 2012; pp. 29–54.

[56] Sands, T. Deterministic Artificial Intelligence. IntechOpen as ISBN: 978-1-78984-111-4.

[57] D. C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Mathematical programming 45 (1989) 503–528.

[58] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals,

Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[59] Azam Asl. Behavior of the Limited-Memory BFGS Method on Nonsmooth Optimization Problems in Theory and Practice. PhD Thesis, New York University. URL: https://cs.nyu.edu/media/publications/asl_thesis_final_UtpoLsu.pdf

[60] Marian P. Kazmierkowski, Mariusz Malinowski, Michael Beach. CHAPTER 4 - Pulse Width Modulation Techniques for Three-Phase Voltage Source Converters. Control in Power Electronics. 2002, Pages 89-160. https://doi.org/10.1016/B978-012402772-5/50005-3

[61] Alberto Ramos. Automatic differentiation for error analysis of Monte Carlo data. Computer Physics Communications. Volume 238, May 2019, Pages 19-35. https://doi.org/10.1016/j.cpc.2018.12.020

[62] A. Pinkus, Approximation theory of the MLP model in neural networks, Acta Numerica, 8. (1999), pp. 143–195.