

Article

Comparison of Deep Learning and Deterministic Algorithms for Control Modeling

Hanfeng Zhai  and Timothy Sands  *¹ Sibley School of Mechanical and Aerospace Engineering, Cornell University, Ithaca, NY 14850, USA² Department of Mechanical Engineering (CVN), Stanford University, Stanford, CA 94305, USA

* Correspondence: dr.timsands@alumni.stanford.edu

Abstract: Controlling nonlinear dynamics arises in various engineering fields. We present efforts to model the forced van der Pol system control using physics-informed neural networks (PINN) compared to benchmark methods, including idealized nonlinear feedforward (FF) control, linearized feedback control (FB), and feedforward-plus-feedback combined (C). The aim is to implement circular trajectories in the state space of the van der Pol system. A designed benchmark problem is used for testing the behavioral differences of the disparate controllers and then investigating controlled schemes and systems of various extents of nonlinearities. All methods exhibit a short initialization accompanying arbitrary initialization points. The feedforward control successfully converges to the desired trajectory, and PINN executes good controls with higher stochasticity observed for higher-order terms based on the phase portraits. In contrast, linearized feedback control and combined feed-forward plus feedback failed. Varying trajectory amplitudes revealed that feed-forward, linearized feedback control, and combined feed-forward plus feedback control all fail for unity nonlinear damping gain. Traditional control methods display a robust fluctuation for higher-order terms. For some various nonlinearities, PINN failed to implement the desired trajectory instead of becoming “trapped” in the phase of small radius, yet idealized nonlinear feedforward successfully implemented controls. PINN generally exhibits lower relative errors for varying targeted trajectories. However, PINN also shows evidently higher computational burden compared with traditional control theory methods, with at least more than 30 times longer control time compared with benchmark idealized nonlinear feed-forward control. This manuscript proposes a comprehensive comparative study for future controller employment considering deterministic and machine learning approaches.



Citation: Zhai, H.; Sands, T. Comparison of Deep Learning and Deterministic Algorithms for Control Modeling. *Sensors* **2022**, *22*, 6362. <https://doi.org/10.3390/s22176362>

Academic Editor: Kyandoghere Kyamakya

Received: 22 July 2022

Accepted: 15 August 2022

Published: 24 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

As early as (at least) the late 19th century, scientists made efforts to design and implement control systems to deal with instability, oscillation, and various nonlinear and chaotic phenomena [1]. Maxwell studied valve flow governors [2], while more recently, Cartwright et al. used the van der Pol equation in seismology to model the two plates in a geological fault [3]. Fitzhugh [4,5] used the equation to model the action potentials of neurons. Systems exhibiting strong nonlinear behavior are tough problems to control. The standard practice of base controls on the linearization of the system is often rendered ineffective due to the elimination of the nonlinear features. Machine learning is one approach with seeming applicability due to its ability to learn and control nonlinear features.

1.1. Physics-Informed Machine Learning

There has been significant recent progress in the field of machine learning in recent decades, starting from the late 80s following the utter failure to achieve its “grandiose objectives” in the 1970s. Ref. [6] Taking advantage of “big data” and advanced computing technologies such as GPU and TPU computing, there has been exponential growth in the

field of deep learning. Central Processing Units (CPU) manage all the functions of a computer and can be augmented by Graphical Processing Units (GPU) and Tensor Processing Units (TPU) to accelerate calculations with application-specific integrated circuits. In the past five years, an explosion of research has re-instantiated “grandiose objectives” manifest in “deep learning”. There have been attempts to insert physical information into neural networks (NN) since at least the 1990s, [7] relying both on statistical and symbolic learning, called hybrid learning [8–11]. Towell et al. [8] described hybrid learning methods using theoretical knowledge of a domain and a set of classified examples to develop a method for accurately classifying examples not seen during training. Towell et al. [9] introduced methods to refine approximately correct knowledge to be used to determine the structure of an artificial neural network and the weights on its links, thereby making the knowledge accessible for modification by neural learning. Towell et al. [10] illustrated a method to efficiently extract symbolic rules from trained neural networks.

Meanwhile, the recent development of physics-informed neural networks (PINNs), originally introduced in 2017 [12], encode differential equations in the losses of the NNs as a soft constraint enabled by automatic differentiation [13], allowing fast, efficient learning of physical mapping with relatively less labeled data. One well-known application is in the field of fluid fields [14,15]. An aspect not well known or studied is the implementation of control signals for nonlinear systems using PINNs enabled by inserting the control signals and positional constraints into the loss. This aspect is known as physics-informed deep operator control (PIDOC) [16]. Particularly, it is shown in this work that PIDOC can successfully implement controls to nonlinear van der Pol systems yet fails to converge to the desired trajectory when the system’s nonlinearity is large.

1.2. Deterministic Algorithms

In 2017, Cooper et al. [17] illustrated how an idealized nonlinear feedforward very effectively controlled highly nonlinear van der Pol systems with fixed parameters, while [16] adopting Cooper’s method as the benchmark for comparison, as done here in this manuscript. Based on the work presented in this manuscript on NN-based control and deterministic algorithms, it can be deduced that challenging problems remain open, particularly regarding controlling highly nonlinear systems. The “grandiose objectives” referred to by Sir Lighthill [6] remain unfulfilled, and this insight guides both industry and academia efforts in controller design and system stability analysis.

There have also been attempts at comparing classical PID controllers with neural networks [18], refining PID controllers with neural networks [19,20], or inserting neural networks into traditional controllers in general [21–23]. Hagan and Demuth [21] provide a quick overview of neural networks and explains how they can be used in control systems. Nguyen et al. [22] demonstrated a neural network could learn of its own accord to control a nonlinear dynamic system, while Antsaklis [23] evaluated whether neural networks can be used to provide better control solutions to old problems or perhaps solutions to control problems that have proved to confound.

Inserting nonlinear approximation by neural networks to refine control and stability is not a new thing and is considered a type of “learning control” dating back to the 80s and 90s. Notwithstanding, as already introduced in [16], building control frameworks solely with neural networks is relatively rare. Acknowledging the deficiency of related works, this manuscript provides a fairly comprehensive analysis of PIDOC [16] as well as the original methods proposed by Cooper et al. [17] on the van der Pol system as a nonlinear representation of oscillating circuits, amongst other example applications. The key question we strive to answer in this paper is: What are the main differences between different methodologies in control modeling of nonlinear dynamics? To answer this, a benchmark is designed considering both the works and analysis of the systematic behavior. Afterward, desired trajectories were modified from the benchmark problem to check how the control methods differ by testing their first and second-order phase portraits.

In this manuscript's Section 2 we briefly formulate the problem with a brief introduction to the van der Pol system and control schemes. In Section 3 we introduce the control approaches, including physics-informed deep operator control (Section 3.1), containing deep learning (Section 3.1.1) and physics-informed control (Section 3.1.2); and control theory algorithms in Section 3.2, with linearized feedback control (Section 3.2.1); idealized nonlinear feed-forward control (Section 3.2.2) and combined control (Section 3.2.3); we then briefly how the methods are compared in Section 3.3. Next Section 4 includes results comparing the control schemes: Section 4.1 shows how the methods differ on the benchmark problem; Section 4.2 shows how changing desired trajectories variate the controlled schemes; Section 4.3 shows how variegating systematic nonlinearity changes different control results.

2. Problem Formulation

As introduced in Section 1, the main goal is a comparison of control methods. The comparison studies are conducted on the van der Pol systems as prototype systems. A general system schematic of this paper is illustrated in Figure 1. The command signal was calculated by the controller, passing the control commands to the system, where the system's nonlinear behavior is sensed and fed back using a sensor (not illustrated in the schematic). As the control loop stabilizes, the controlled dynamics are output for real-world applications. This manuscript mainly focuses on the controller (red box in Figure 1), PIDOC, and other control methods are all codified in the controller box.

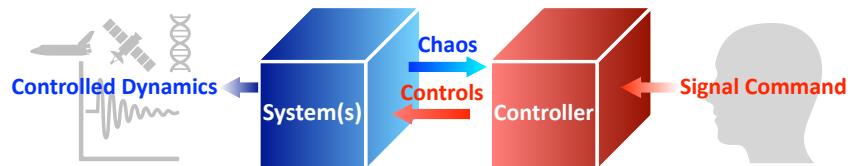


Figure 1. A basic schematic diagram for a control process. The human desired signal command is input to the system through the controller as illustrated in the red box, which passes the control to the targeted system in a “feedforward-feedback-control” loop. Note that the “chaos” from the systems, as in the blue box, is passed to the controller through the sensor. The final controlled dynamics are output to different applications as illustrated in the left schematic marked as “controlled dynamics”. Detailed description, please see text.

The van der Pol system was adopted to test the control signals' implementation, and a phase portrait of the van der Pol system is illustrated in Figure 2 where the system is arbitrarily initialized. Given arbitrary initial points, the trajectory always becomes “entrapped” in a nonlinear track (called a limit cycle), while control methods strive to release the trajectory from the trapped path along the limit cycle and drive the trajectory to some desired, commanded behavior. Such a system was first discovered by van der Pol when investigating oscillating circuits, taking the form [24,25]. van der Pol [24] introduced an oscillatory system with a damping that is negative. Together with van der Mark [25], he also illustrated how to design an electrical system such that alternating currents or potential differences will occur in the system, having a frequency that is a whole multiple of the forcing function.

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

where in the original circuits formulation, $x(t)$ is the current measured in amperes as the rate of change of the charge [26] and μ is a scalar parameter indicating the nonlinearity and the strength of the negative damping [16]. Henceforth, $x(t)$ is referred to as position.

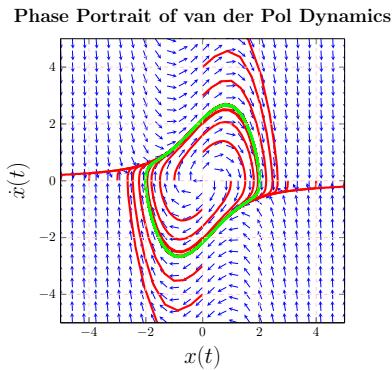


Figure 2. The inherent dynamics of the van der Pol equation [27]. The light green line indicates the limit cycle, the manifestation of the strong nonlinearity of the van der Pol inherent dynamics. Disparate red lines indicate trajectories beginning at various initial points, which all eventually fall onto the inherent limit cycle. The blue arrows indicate the total phase field of the van der Pol inherent dynamics, indicating the Towe directions.

For testing the proposed methods, control signals are formulated and passed forward to the nonlinear system as commands. The simulated system duplicated the system introduced in [16], where the MATLAB command `odeint` solves the equations providing data to feed the training of PIDOC. The van der Pol equation was solved in the time domain from time, $t = [0, 50]$, and interpolated with 5000 points. The error control parameters `rtol` and `atol` are 10^{-6} and 10^{-10} , respectively [28].

3. Methodology and Materials

This section briefly outlines the theoretical foundation of the physics-informed neural network-based algorithm and the alternative based on traditional control theory. The methodology of subsequent numerical experiments used for testing the methods is also introduced.

3.1. Physics-Informed Deep Operator Control

3.1.1. Deep Learning

Physics-informed deep operator control is enabled by the general deep neural network framework, where for the van der Pol system, the position is inferred based on the input time domain in accordance with Equation (2). Given an input time series t , the “predicted” trajectories out of the neural networks are x_{pred} . The process can be symbolized as

$$x_{pred} = (K_L \circ \sigma_L \circ \dots \circ K_1 \circ \sigma_1 \circ K_0)t \quad (2)$$

where K_1, K_2, \dots, K_L are linear layers; $\sigma_1, \sigma_2, \dots, \sigma_L$ are the activation functions, where PIDOC employs \tanh activation functions. More details of such methods can be found in Zhai and Sands [16].

A supervised machine learning framework is defined using external training data as a formulation minimizing the loss function so that the neural network can capture data features through an optimization process, whereas in traditional neural network approaches \mathcal{L} is usually the difference (errors) between the neural network predictions and training data. Let $\mathcal{L} = \mathcal{L}(t, \mathbf{p})$ denote the loss function, where t is the input time series and \mathbf{p} is the parameter vector contained in formations of \mathcal{I} , \mathcal{D} , and neural network. As no external constraints or bounds are enforced, the optimization problem hence takes the form of Equation (3) [16].

$$\min_{t \in \mathbb{R}^{d_{out}}} \mathcal{L}(t, \mathbf{p}) \quad (3)$$

Minimizing \mathcal{L} requires reiterating the neural network as defined for the “training”. The limited-memory Broyden–Fletcher–Goldfarb–Shanno optimization algorithm, a quasi-

Newton method (L-BFGS-B in TensorFlow 1.x) [29,30] is adopted. Optimization is carried over iterations looping from the blue box (neural network) to purple box (\mathcal{I} & \mathcal{D}) to red box (\mathcal{L}) displayed in Figure 3. 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 3.1.2.

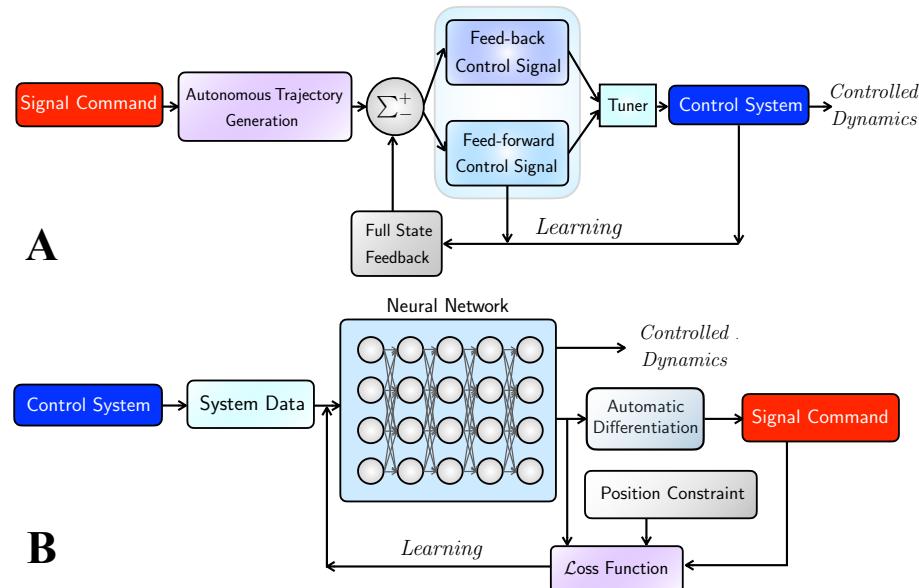


Figure 3. Schematic diagram for the deterministic control algorithms and the deep learning-based PIDOC control scheme. (A) The schematic for deterministic control algorithms. Note that the light blue tuner can switch the algorithms either to pure idealized nonlinear feed-forward (symbolized as \mathcal{FF} , as illustrated in the bottom blue box), linearized feedback (symbolized as \mathcal{FB} , as illustrated in the upper dark blue box), or the combined control scheme (symbolized as \mathcal{C} , combined both \mathcal{FF} and \mathcal{FB}). (B) The schematic for PHYSICS-INFORMED DEEP OPERATOR CONTROL (PIDOC), symbolized as PID , where the control signal \mathcal{D} (represented in the red box) is inserted in the loss function \mathcal{L} in the purple box as part of the PINN. Detailed description, please see the text.

3.1.2. Physics-Informed Control

According to reference [16], the control function is enabled by encoding the control signal into the loss function of the neural network, inspired by the formulated physics-informed neural networks (PINNs) [12], where the loss function is computed through the mean squared errors (MSE) elaborated in Equation (4).

$$\mathcal{L} = MSE_{NN} + MSE_{\mathcal{I}} + MSE_{\mathcal{D}} \quad (4)$$

where MSE_{NN} , $MSE_{\mathcal{I}}$, $MSE_{\mathcal{D}}$ stands for the neural network generation errors, the initial position loss, and the control signal loss, respectively, computed as Equation (5).

$$\begin{aligned} MSE_{NN} &:= \frac{1}{N} \sum_{i=1}^N \left[x_{train} - x_{pred} \right]^2 \\ MSE_{\mathcal{I}} &:= \frac{1}{N} \sum_{i=1}^N \left[x_{pred}^0 - x_{\mathcal{D}}^0 \right]^2 \\ MSE_{\mathcal{D}} &:= \frac{1}{N} \sum_{i=1}^N \left[\left(\frac{dx_{\mathcal{D}}^2}{dt^2} - \frac{dx_{pred}^2}{dt^2} \right) + \left(x_{\mathcal{D}} - x_{pred} \right) \right]^2 \end{aligned} \quad (5)$$

where $x_{\mathcal{D}}^0$ denotes the initial position of desired trajectory; x_{pred} is the neural network predicted output; x_{train} is the given training data (from system simulation); x_{pred}^0 and $x_{\mathcal{D}}^0$

denote the initial positions of the neural network predicted output and desired trajectory. Detailed formulations are elaborated by reference [16].

To impose the triangular function signals, we simply impose the form of x_D in Equation (6).

$$x_D(t) = \Lambda \sin(t), \implies \dot{x}_D(t) = \Lambda \cos(t), \ddot{x}_D(t) = -\Lambda \sin(t) \quad (6)$$

Based on such an x_D , the output phase portrait ($\dot{x}(t)$ versus $x(t)$ phase portrait) is expected to be a circular trajectory. To implement different amplitudes of the desired trajectory Λ , we modify Equation (5) to encode the amplitude information into the neural network losses, given the same training data resulting in Equation (7).

$$MSE_{NN} := \frac{1}{N} \sum_{i=1}^N \left[x_{train} - \frac{x_{pred}}{\Lambda} \right]^2 \quad (7)$$

where the above equations represent the general formulation of PIDOC. The detailed graphical representation is illustrated as in Figure 3B: the control system (deep blue box) first generates nonlinear data that feeds into the neural network, forwards the output to encode the control signals as shown in the deep red box into the loss function through automatic differentiation, and reiterates the training of the neural network until the control signal is fine-tuned for systematic output.

3.2. Deterministic Control Algorithms

For the alternative application of control theory, the general framework begins with the modification of Equation (8), where controller gains are calculated through the Riccati equation becoming a controller known as the linear quadratic regulator (LQR) [17].

$$\frac{d^2x}{dt^2} - \mu(1 - x^2) \frac{dx}{dt} + x = F(t) \quad (8)$$

where $F(t)$ is forced on the nonlinear systems to exert control. By modifying $F(t)$, different types of controls are implemented, where in our approach, we adopt nonlinear feed-forward (\mathcal{FF}), linearized feedback control (\mathcal{FB}), and the combined controls, to be elaborated in Sections 3.2.1–3.2.3, respectively.

3.2.1. Linearized Feedback Control

In control theory and sciences, a common first step in control design is linearizing nonlinear dynamic equations and then designing the control based on that linearization. For the van der Pol dynamics, Equation (8) can be linearized and reduced into Equation (9), expressed in state-variable formulation from which state space trajectories are displayed on phase portraits [17].

$$\frac{dx}{dt} = \mathbf{Ax} + \mathbf{Bu} \quad (9)$$

The infinite-horizon cost function given by Equation (10)

$$J = \int_0^{t_{end}} [x^\top \mathbf{Q} x + u^\top \mathbf{R} u] dt, \quad \mathbf{Q} = \mathbf{Q}^\top \succeq 0, \mathbf{R} = \mathbf{R}^\top \succ 0 \quad (10)$$

The goal is to find the optimal cost-to-go function $J^*(x)$ which satisfies the Hamilton–Jacobi–Bellman Equation (11)

$$\forall x, 0 = \min_u \left[x^\top \mathbf{Q} x + u^\top \mathbf{R} u + \frac{\partial J^*}{\partial x} (\mathbf{Ax} + \mathbf{Bu}) \right] \quad (11)$$

where to find solutions, Equation (12) is formed necessitating solution of (13) which is the *algebraic Riccati equation*. The solution of the equation is of well-known form. Note

that the computation of K_p , K_d , and $[S]$ are based on MATLAB® command $[K, S, E] = \text{lqr}(A, B, Q, R)$

$$J^*(x) = x^T S x, \quad S = S^T \succeq 0 \quad (12)$$

$$0 = SA + A^T S - SBR^{-1}B^T S + Q \quad (13)$$

where A and B are the expressions used in the linear-quadratic optimization leading to a feedback controller with linear-quadratic optimal proportional and derivative gains for K_p and K_d . The closed loop dynamics are established by Equation (14) where the van der Pol forcing function $F(t)$ is a proportional-derivative (PD) controller whose gains used in this manuscript are from [17].

Adopting the linearized feedback control by Cooper et al. [17], Equation (8) can thence be expanded in the form:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2) \frac{dx}{dt} + x \equiv F_{FB}(t) = -K_d(\dot{x}_d - \dot{x}) - K_p(x_d - x) \quad (14)$$

where x_d is the desired trajectory; K_d and K_p are the derivative and proportional gain, respectively. Similar with our approach in Equation (6), x_d is the desired control trajectory, writes $x_d = \Lambda \sin(t)$.

3.2.2. Nonlinear Feed-Forward Control

In idealized nonlinear feed-forward controls, the forced term $F(t) = F_{FF}(t)$ having the form of the original van der Pol system with the desired trajectory $x = x_d$ executed on:

$$\frac{d^2x}{dt^2} - \mu(1 - x^2) \frac{dx}{dt} + x \equiv F_{FF}(t) = \frac{d^2x_d}{dt^2} - \mu(1 - x_d^2) \frac{dx_d}{dt} + x_d \quad (15)$$

where x_d is the desired signal, as in Equation (14). By implementing x_d in the force term, the control is thence applied to the van der Pol system, defined as the nonlinear feed-forward control as the executed force term possesses the form of idealized nonlinear trajectory.

3.2.3. Combined Control

To apply both the idealized nonlinear feedforward trajectory combined with the linearized feedback, the force term of the combined control simply follows

$$F_C(t) = F_{FF}(t) + F_{FB}(t) \quad (16)$$

where F_{FB} and F_{FF} are elaborated in Equations (13) and (14), respectively. F_C is then applied to the van der Pol system in following the same form as in Equations (13) and (14).

The basic framework of the controls is shown in Figure 3A: the signal command as shown in the deep red box (x_d in our equations) is the first input to the automatic trajectory generator that is forwarded to the gains, and then forwarded to either feed-forward controls (F_{FF}) on the lower light blue box or feedback controls (F_{FB}) on the upper dark blue box or the combined approach. The control signals are tuned through the light blue tuner box on the right, which controls the force term applied to the nonlinear system as indicated in the solid blue box on the right. After exerting the desired control signals, the output signals are first fed to the gains as full state feedback indicated in the gray box; the final controlled dynamics are output after the workflow is executed iteratively.

3.3. Comparison and Estimation

To conduct a fair, decent, and comprehensive comparison of the proposed methods, we consider *Systematic analysis* of the provided benchmark problem as it has been mentioned in Section 1, *Trajectory convergence* for different amplitudes of desired trajectories, signified by Λ in Equation (6) and *Non-linearity* of the systems with different nonlinearities, signified

through μ ion Equation (1). For the benchmark systematic behavior analysis, considering both the work of Zhai & Sands [16] and Cooper et al. [17], we pick $\Lambda = 5, \mu = 1$, as a system with low nonlinearity; in which for the PIDOC framework, the NN has the structure of 6×30 . The initial point is picked as (1,0). For systems of different desired amplitudes, Λ is changed from 1, 3, 5, 7, 9. For systems of different non-linearities, μ is changed from 1, 3, 5, 7, 9, 10. The PIDOC was conducted in Google Colab [31] using Python 3.6 compiling TensorFlow 1.x [30]. Both \mathcal{FF} , \mathcal{FB} and \mathcal{C} were written in MATLAB R2021A and executed with Simulink.

4. Results and Discussion

4.1. Benchmark Analysis

The results of the benchmark analysis are shown in Figure 4, where Figure 4A,B stand for the first and second order phase portraits of different controlled schemes by PIDOC, \mathcal{FF} , \mathcal{FB} , & \mathcal{C} , marked in different colors dashed lines as elaborated in the caption; compared with the inherent van der Pol dynamics and desired trajectory marked in black and pink solid lines, respectively. The desired trajectories marked in pink are the same as previous works in the field [32–34]. The phase Figure 4C,D illustrated the time evolution of the zeroth, first, and second order derivatives of the position $x(t)$, with the same color representations as in Figure 4A,B. Given the benchmark problem, it can be deduced that all the control theory methods exhibit strong fluctuations at the initial stage of controls, where \mathcal{FF} converge to the desired trajectory successfully, as indicated in the deep blue dashed lines, whereas both \mathcal{FB} and \mathcal{C} fails. Another interesting point to be noted is that all the traditional control algorithms exhibit a stronger fluctuation for higher order terms at the beginning stage, yet \mathcal{FF} successfully converge to the trajectory that exhibits better control effects than PIDOC, but \mathcal{FB} and \mathcal{C} displays such a robust fluctuation along the time. To this phenomenon, we provide the following explanation: the errors generated by the linearization of the van der Pol equation accumulate and cause the robust fluctuations as indicated in Figure 4 for the light blue and red lines. However, admittedly, \mathcal{FF} successfully implements the control with higher accuracy for higher order terms than PIDOC; but noted that as \mathcal{FF} only forwarding control signals can be considered as an open-loop system, in real-world practice, trivial noises will be accumulated that leads to the in-feasibility of \mathcal{FF} .

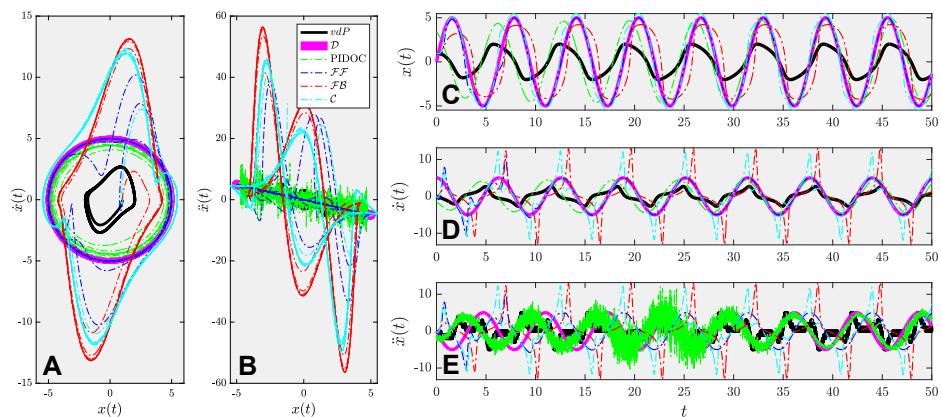


Figure 4. System behavior analysis for the benchmark problem. Note the inherent van der Pol dynamics (vdp) is marked in black solid line; the desired trajectory (\mathcal{D}) is marked in pink solid line; the PIDOC control is marked in light green dashed line; the feed-forward control (\mathcal{FF}) is marked in dark blue dashed line; the feedback control (\mathcal{FB}) is marked in red dashed line; the combined control (\mathcal{C}) is marked in the light blue line. (A) The phase portrait of the van der Pol systems of inherent dynamics, desired trajectory, and different control schemes marked in different colors. (B) The acceleration-position plot. (C) The time evolution of positions. (D) The time evolution of velocities. (E) the time evolution of accelerations.

4.2. Trajectory Amplitude

The results of controlled dynamics of trajectories of the first and second-order phase portraits are shown in Figures 5 and 6, respectively. It can be discerned from Figure 5A,B that both PIDOC (symbolized as $\Pi\mathcal{D}$ in the figure) and \mathcal{FF} are able to implement controls with an exception of **B1** that \mathcal{FF} failed to control the system when $\Lambda = 1$. Similar to the benchmark problem that both \mathcal{FB} and \mathcal{C} failed to implement the controls with a highly fluctuating behavior, in Figure 5C,D. An interesting phenomenon reported from **D1** to **D5** is that with increasing trajectory amplitudes we report a better convergence for the combined (\mathcal{C}) control. We can hence propose the discussion on such phenomena that for higher values of desired trajectory amplitudes, the linearization effect of the feedback reduces for the van der Pol systems.

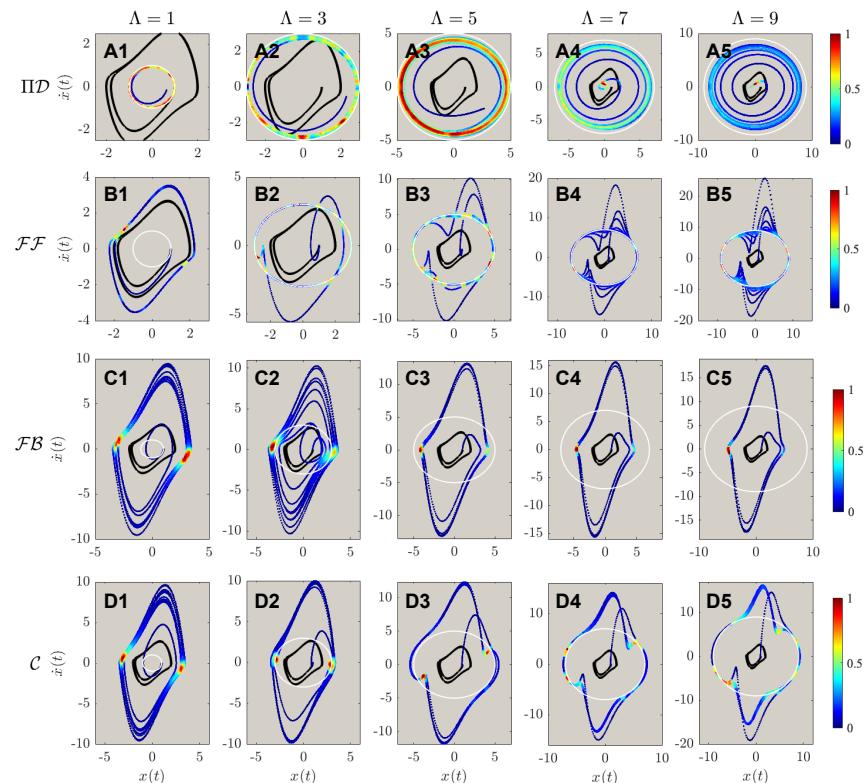


Figure 5. The phase portrait of different controlled schemes, including the inherent van der Pol dynamics, marked in solid black line; the desired trajectory marked in white dashed line; and the controlled dynamics marked in the contoured line. The contour legend was marked from 0 to 1, showing the intensity of the controls. (A1–A5) shows the controls by physics-informed deep operator control, symbolized by $\Pi\mathcal{D}$, of different desired trajectories from $\Lambda = 1, 3, \dots, 9$. (B1–B5) shows the controls by feed-forward controls (\mathcal{FF}), from $\Lambda = 1, 3, \dots, 9$. (C1–C5) shows the controls by feedback controls (\mathcal{FB}), from $\Lambda = 1, 3, \dots, 9$. (D1–D5) shows the controls by feed-forward - feedback combined controls (\mathcal{C}), from $\Lambda = 1, 3, \dots, 9$. It can be observed that for $\Pi\mathcal{D}$, the controlled states achieved good accuracy of different Λ ; \mathcal{FF} archives good control qualities except $\Lambda = 1$; both \mathcal{FB} and \mathcal{C} are reported to not able to implement the circular controlled trajectories into the inherent dynamics.

Figure 6 reports the second order phase portraits (acceleration-position diagram) comparing the four methods. Figure 6A reports the stochastic approximation nature of PIDOC: the learning-based control executes control signals based on randomized sampling for trajectory convergence. Corresponds to Figure 5B1 shows the failure of \mathcal{FF} control when $\Lambda = 1$; whereas **B2** to **B5** shows how the second order phase portraits display a higher fluctuation, as also shown from in Figure 6C,D. Figure 6B also shows a strong discretized form of \mathcal{FF} control, as illustrated based on the sparse points. The control contour from

both Figures 5 and 6 both \mathcal{FB} and \mathcal{C} controls (sub-figure **C** and **D**) shows an increased control density on the horizontal edges ($x(t)$ direction), indicated by the denser points.

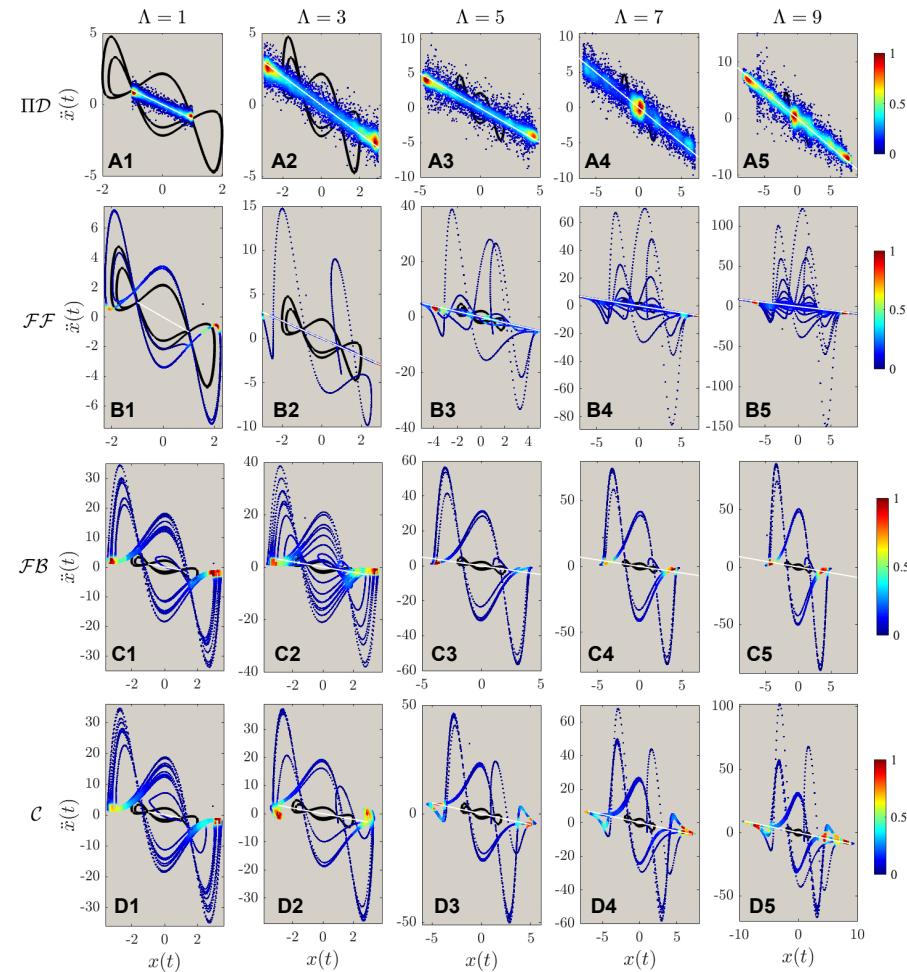


Figure 6. The acceleration-position portrait of different controlled schemes, with the marking colors, is the same as in Figure 5. Note that (A1–D5) are the same as in Figure 5: the implementations of IID, \mathcal{FF} , \mathcal{FB} , & \mathcal{C} to different targeted trajectory amplitudes of $\Lambda = 1, 3, \dots, 9$. The second order phase portrait shows that the deterministic algorithms (\mathcal{FF} , \mathcal{FB} , \mathcal{C}) exhibit higher fluctuations during the control procedures, with higher fluctuation values corresponding to higher targeted circular trajectories' radii.

The total computational burden of the four methods is shown in Table 1: the PIDOC framework shows an evidently larger computing time than \mathcal{FF} , \mathcal{FB} and \mathcal{C} ; generally, \mathcal{FF} execute the fastest control and \mathcal{C} exhibits the longest control time within the tested control theory algorithms. We provide the following explanations for the above phenomena: (1) the PIDOC framework is based on the training of the NN, where the approximation of nonlinear data takes exponentially longer compared with just implementing the control commands; (2) since \mathcal{FF} can be considered as an open-loop implementation of control signals, where the elimination of feedback and error adjustment reducing computation time; (3) the combination of both feed-forward and feedback requires estimation of the route execution and linearizations, consumes more time. Based on the computation time one can discern that although more stable control implementations are exhibited by PIDOC, the drawback is also evident: the considerably longer training time required for implementing the control.

Table 1. The computational burden of the four different control frameworks considering different desired trajectories (desired radius Λ). Note that the unit are in seconds (default of `timeit.time()` in Python and `cputime` in MATLAB).

Λ	1	3	5	7	9
$\Pi\mathcal{D}$	329.82	618.97	713.30	261.37	480.77
\mathcal{FF}	5.70	6.01	5.26	6.21	5.66
\mathcal{FB}	6.68	7.23	5.99	6.07	6.87
\mathcal{C}	8.35	6.63	7.62	6.46	6.06

4.3. Nonlinear Effects

The results of different control for systems of different nonlinearities with a fixed desired trajectory $\Lambda = 5$ are shown in Figure 7. Same as reported by Zhai & Sands [16], the PIDOC control fails to implement control for systems of high nonlinearities as to be “trapped” in a smaller radius trajectory. The \mathcal{FF} control was implemented successfully, with a strong fluctuation reported for high nonlinearities observed from **B1** to **B5**, with the failed implementation when $\mu = 10$ as shown in **B6**, which can be considered as nonlinearity threshold. Both \mathcal{FB} and \mathcal{C} also failed for control execution same as in Figures 5 and 6. To note, both the control theory methods implemented show an evident higher data density along the horizontal edges, which can be adopted to infer the nature of control theory methods: stronger control imposition near edges, corresponding to the wave crests and troughs as for the time evolution of the position.

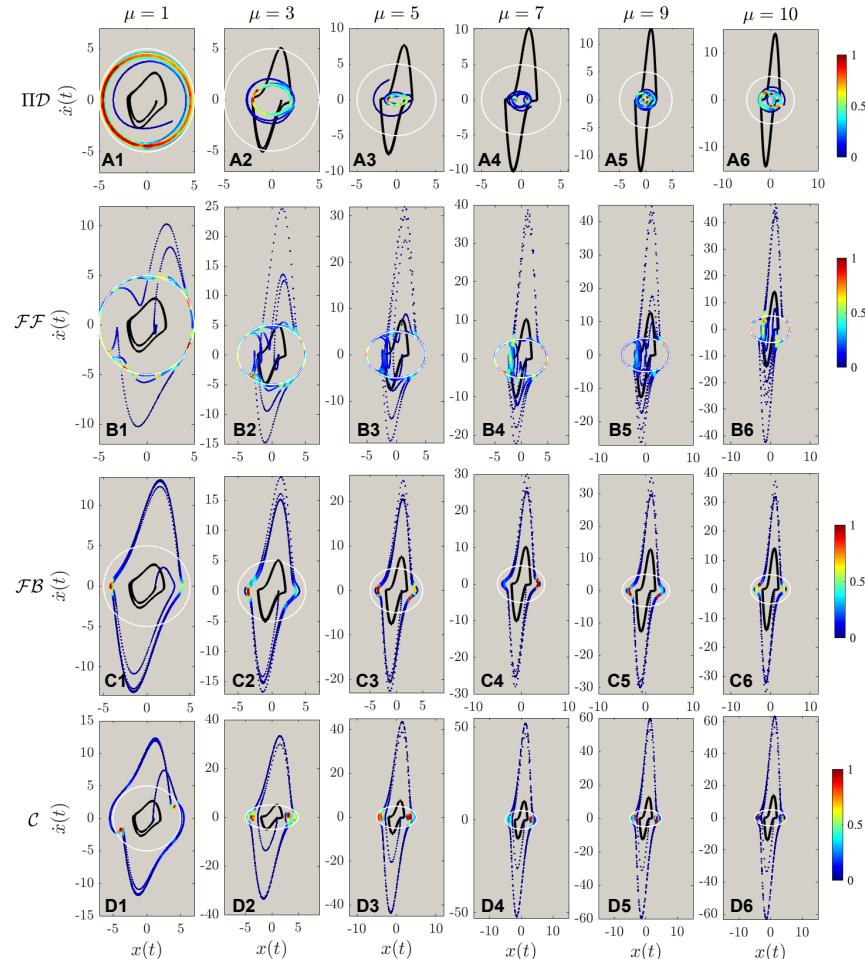


Figure 7. The phase portrait of different controlled schemes, with the marking colors, is the same as in Figure 5. (**A1–A6**) shows the controls by physics-informed deep operator control, symbolized by $\Pi\mathcal{D}$, of different van der Pol systems with different nonlinearities from $\mu = 1, 3, 5, 7, 9, 10$. (**B1–B6**) shows

the controls by feed-forward controls (\mathcal{FF}), from $\mu = 1, 3, 5, 7, 9, 10$. (**C1–C6**) shows the controls by feedback controls (\mathcal{FB}), from $\mu = 1, 3, 5, 7, 9, 10$. (**D1–D6**) shows the controls by feed-forward—feedback combined controls (\mathcal{C}), from $\mu = 1, 3, 5, 7, 9, 10$. It can be observed that $\Pi\mathcal{D}$ fails to implement control for van der Pol systems of higher nonlinearities. Nonlinearities elicit stronger fluctuations compared with Figure 5 for deterministic control algorithms. \mathcal{FF} is reported to have the best control qualities for highly nonlinear systems.

The second order phase portraits are shown in Figure 8: as for the control theory methods, evidently higher nonlinearities are observed for \mathcal{C} compared with \mathcal{FF} and \mathcal{FB} ; a more discrete points distribution indicates larger steps for control implementations. Just by observing Figure 8A, it is discerned that the systematic nonlinearity was very high, as indicated in the solid black line compared with the white dashed line for the desired trajectory. However, comparing **B** to **D** it is observed that for systems of higher nonlinearities, the control displays extremely strong fluctuations at the beginning stage of the control. Based on such a phenomenon, we hence deduce another finding for control theory properties: the control implementation will enlarge the nonlinear signals with larger steps of control discretization. To present a more detailed analysis of Figures 7 and 8, Figure 9 is created for a zoomed view of the control schemes for both first and second-order phase portraits. Interestingly, vortex-like structures are observed in the first-order phase portrait for both PIDOC and \mathcal{C} along the edges of the circular trajectory. Figure 9B6 shows how \mathcal{FF} fails control imposition in detail: an oscillation along the circular causes the “split” of the controlled trajectory vertically, where such a trend has already been observed in Figure 9B5. Figure 9C clarifies a phenomenon that has already been observed and discussed: an increased data density along the edges of the desired control schemes indicates a stronger control implementation along the edges.

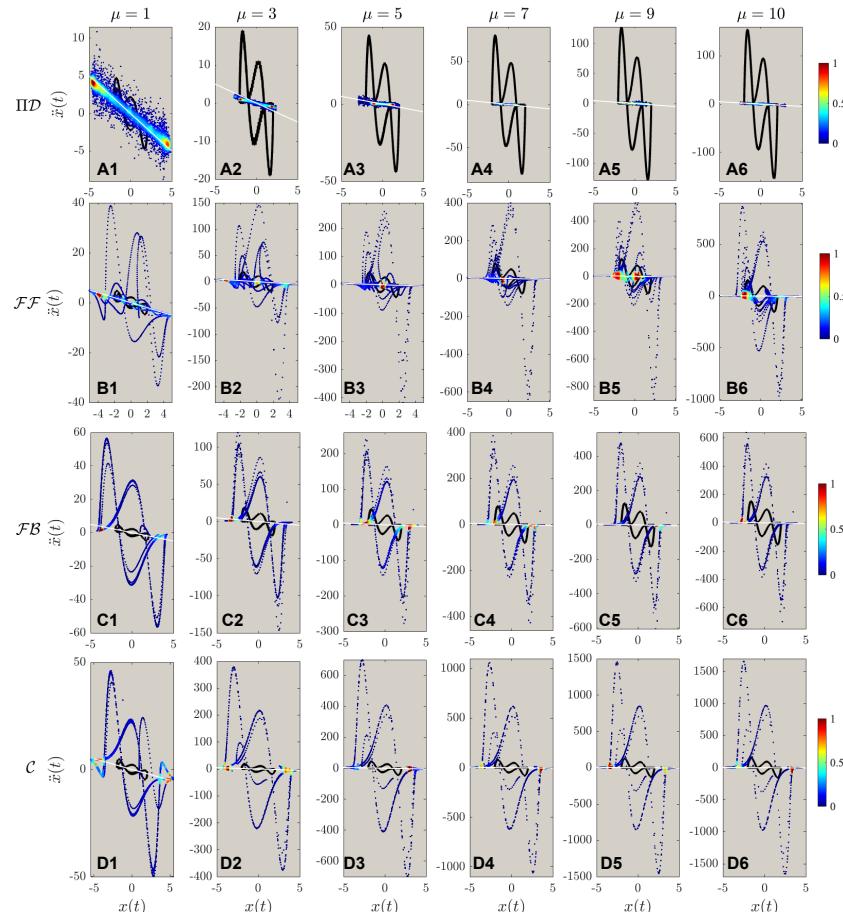


Figure 8. The acceleration-position portrait of different controlled schemes, with the marking colors

same as in Figure 5. Note that (A1–D6) are the same as in Figure 5: the implementations of $\Pi\mathcal{D}$, \mathcal{FF} , \mathcal{FB} , & \mathcal{C} to different targeted trajectory amplitudes of $\Lambda = 1, 3, \dots, 9$. It can be observed that the fluctuations for deterministic algorithms are evidently higher for $\Pi\mathcal{D}$ in the second-order phase portraits. The fluctuations during the control processes grow evidently with regards to increasing nonlinearities.

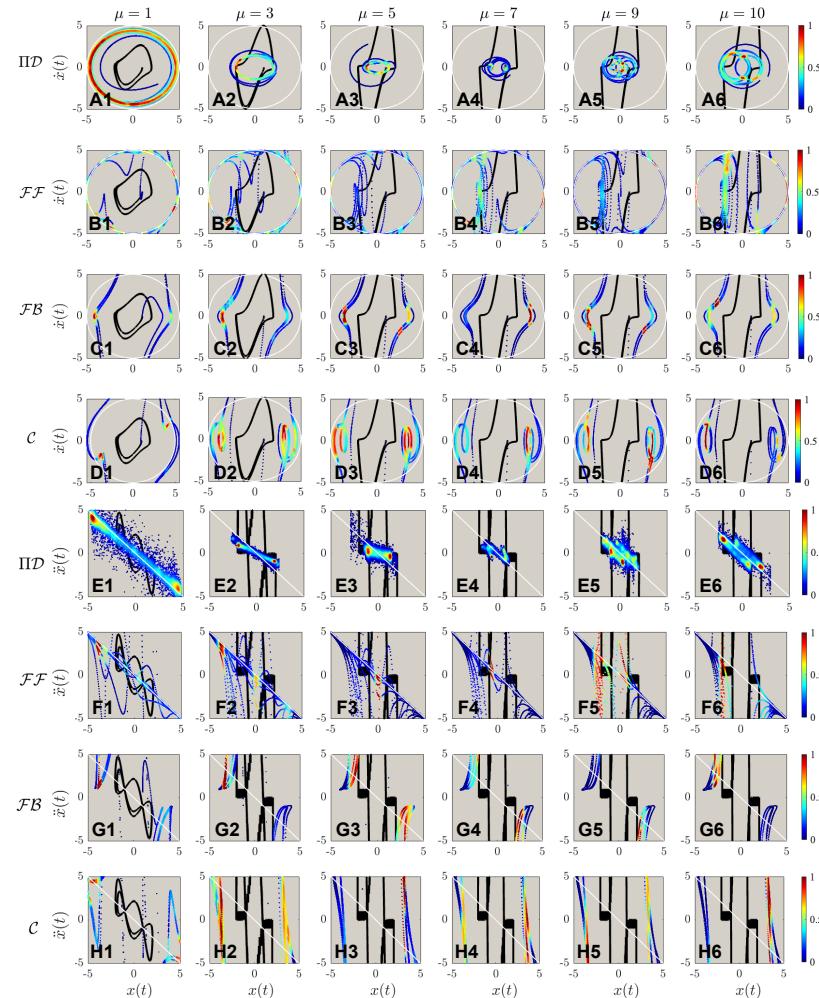


Figure 9. The zoomed view of the controlled schemes portrait corresponding to both Figures 7 and 8. Note that the sub-figures (A1–D6) are the same as in Figure 7; whereas the sub-figures (E1–H6) corresponds to sub-figures (A1–D6) in Figure 8. The zoomed views show the details of control implementations: for $\Pi\mathcal{D}$, the controlled schemes are trapped in smaller trajectories that are “twisting” around the original van der Pol dynamics regime; There are two small “twisted daughter circles” occurred at the two sides of the van der Pol inherent dynamics with higher nonlinearities for both \mathcal{FF} , \mathcal{FB} , and \mathcal{C} .

The computational burden as shown in Table 2 displays similar trends as in Table 1: PIDOC exhibits an evidently higher computation time attributed to the NN training. \mathcal{C} exhibits a higher control time than \mathcal{FF} and \mathcal{FB} . Another interesting phenomenon is: that with the increasing system nonlinearity, PIDOC shows a decreasing computation time. Corresponds to Figures 7–9, we propose the following explanation: as the PIDOC-controlled schemes are entrapped in a trajectory with a lower radius, the NN straining stops at an earlier stage since the optimizer (L-BFGS-B) “discern” that more iterations won’t keep decreasing the loss, which leads to a lower computation time but lower quality control. To better quantify the computational burden differences, Table 3 is created taking nonlinear feed-forward control employed by Cooper et al. [17] as a benchmark: PIDOC

displays evidently higher computational burden compared with \mathcal{FF} , with at least more than 30 times of the benchmark time to up to 100 plus more times.

Table 2. The computational burden of the four different control frameworks considering different systems of nonlinearities (different μ values). Note that the unit is in seconds (same as in Table 1).

μ	1	3	5	7	9	10
$\Pi\mathcal{D}$	713.30	257.64	305.91	225.15	197.76	199.52
\mathcal{FF}	5.26	10.33	7.77	6.45	5.38	5.12
\mathcal{FB}	5.99	5.61	6.84	6.02	5.52	5.30
\mathcal{C}	7.62	5.94	5.12	5.83	5.26	5.42

Table 3. The relative computation time comparing PIDOC and control theory algorithms regarding different trajectories and nonlinearities.

$\hat{\tau}$	$\Pi\mathcal{D}$	\mathcal{FF}	\mathcal{FB}	\mathcal{C}
$\Lambda = 1$	39.4999	1.0000	0.8000	0.6826
$\Lambda = 3$	93.3585	1.0000	1.0905	0.9065
$\Lambda = 5$	93.6090	1.0000	0.7861	0.6903
$\Lambda = 7$	40.4602	1.0000	0.9396	0.9613
$\Lambda = 9$	79.3355	1.0000	1.1337	0.9340
$\mu = 1$	135.6084	1.0000	1.1388	1.4487
$\mu = 3$	51.5006	1.0000	0.9444	1.7391
$\mu = 5$	38.6258	1.0000	1.3359	1.5176
$\mu = 7$	34.2228	1.0000	1.0326	1.1063
$\mu = 9$	42.8036	1.0000	1.0494	1.0228
$\mu = 10$	47.5353	1.0000	0.9779	0.9446

To quantify the control errors, Table 4 is generated to compare the control qualities based on the absolute errors. The equation for computing the average absolute relative errors of different control signals are

$$\|\hat{\mathcal{E}}\| = \sum_{i=1}^M \frac{1}{M} \left\| \frac{x_{control} - x_{\mathcal{D}}}{x_{\mathcal{D}}} \right\| \quad (17)$$

It can be observed from Table 4 that PIDOC generally exhibits lower control errors compared with traditional control methods in different trajectories. For different nonlinearities, corresponding to Figure 9, it can be observed that nonlinear idealized feed-forward control exhibits better control qualities.

Table 4. The average absolute relative errors computed from the Equation (17) quantifying the control errors in correspondence with Figures 5 and 7.

$\ \hat{\mathcal{E}}\ $	\mathcal{C}	\mathcal{FF}	\mathcal{FB}	$\Pi\mathcal{D}$
$\Lambda = 1$	2.1379	1.7199	2.0618	0.2225
$\Lambda = 3$	0.3645	0.4124	0.4473	0.2102
$\Lambda = 5$	0.3884	0.4245	0.6694	0.2128
$\Lambda = 7$	0.4168	0.4260	0.7288	0.3387
$\Lambda = 9$	0.4232	0.4264	0.7408	0.2788
$\mu = 1$	0.3884	0.4245	0.6694	0.2056
$\mu = 3$	0.8889	0.4306	0.6327	0.6590
$\mu = 5$	0.8819	0.4353	0.6387	0.6074
$\mu = 7$	0.8782	0.4425	0.6432	0.6345
$\mu = 9$	0.8757	0.4443	0.6466	0.7101
$\mu = 10$	0.8748	0.4847	0.6481	0.6690

5. Conclusions

The nonlinear dynamics control modeling problems of the van der Pol system are tackled by comparing deep learning with traditional deterministic algorithms in this paper. The key idea of this work is to elaborate on the main differences by conducting a comprehensive comparison and benchmark for the recently proposed physics-informed neural networks control with other deterministic algorithms. We first design a benchmark problem for testing the system response for different methods. The desired trajectory and systematic nonlinearity are then changed to check the systematic responses of different controls. The computation burdens are also considered for different methods.

For benchmark analysis, results indicated that all the control theory algorithms exhibit a strong fluctuation which can be interpreted as enlarging the nonlinear inherent van der Pol dynamics with \mathcal{FF} successfully implementing the controls, but the rest fails. The “nonlinearity enlargement” effect is observed to be more obvious for higher order terms. The PIDOC exhibits stochastic nature, which can be attributed to the nature of deep learning inference, same as reported by Zhai & Sands [16]. When changing the trajectory amplitudes, an interesting phenomenon is that \mathcal{FF} failed for trajectory convergence when $\Lambda = 1$. Also, a higher control signal implementation density is observed along the horizontal edges of the first order phase portraits, unveiling control theory imposition to van der Pol systems executes stronger controls along the “signal waves’ crest and trough.” An evidently higher computation burden is observed for PIDOC in comparison to control theory methods. We explain such by the nature of NN learning: the recursive randomization of the NN weights and biases took a longer time than the direct execution of the control signal. For the van der Pol systems with different nonlinearities, it is observed that \mathcal{FF} fails the control when $\mu = 10$, whereas PIDOC also failed to implement controls when $\mu \neq 1$, as the controlled schemes were “trapped” into smaller trajectories. The “nonlinearity enlargement effect” for higher-order phase portraits for control theory algorithms. An interesting phenomenon of a vortex-liked structure of the controlled schemes, as originally reported by Zhai & Sands [16], has also been reported for the \mathcal{C} controls. The evidently higher computation time is also reported for PIDOC, the same as what has been reported for different trajectories. For PIDOC, the computation burden generally reduces with systems of higher nonlinearities. The proposed comparison can guide the future implementation of deep learning-based controller designs and industrial selections.

Author Contributions: Conceptualization, H.Z. and T.S.; methodology, H.Z. and T.S.; software, H.Z.; validation, H.Z. and T.S.; formal analysis, H.Z.; investigation, H.Z. and T.S.; resources, T.S.; data curation, H.Z.; writing—original draft preparation, H.Z.; writing—review and editing, H.Z. and T.S.; visualization, H.Z.; supervision, T.S.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding. The APC was funded by T.S.

Data Availability Statement: All the data and code will be made publicly available upon acceptance of the manuscript through <https://github.com/hanfengzhai/PIDOC> (accessed on 14 August 2022). The Simulink file for the deterministic control methods is available upon reasonable requests to the corresponding author. The Simulink file was originally published by Cooper [17].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Routh, E.J. *A Treatise on the Stability of a Given State of Motion, Particularly Steady Motion: Particularly Steady Motion*; Macmillan and Co.: Cambridge, UK, 1877.
2. Maxwell, J.C. On Governors. *Proc. R. Soc.* **1868**, *100*, 270–283. [[CrossRef](#)]
3. Cartwright, J.; Eguíluz, V.; Hernández-García, E.; Piro, O. Dynamics of Elastic Excitable Media. *Int. J. Bif. Chaos* **1999**, *9*, 2197–2202. [[CrossRef](#)]
4. FitzHugh, R. Impulses and Physiological States in Theoretical Models of Nerve Membrane. *Bio. J.* **1961**, *1*, 445–466. Available online: [https://www.cell.com/biophysj/pdf/S0006-3495\(61\)86902-6.pdf](https://www.cell.com/biophysj/pdf/S0006-3495(61)86902-6.pdf) (accessed on 14 August 2022). [[CrossRef](#)]

5. Nagumo, J.; Arimoto, S.; Yoshizawa, S. An Active Pulse Transmission Line Simulating Nerve Axon. *Proc. IRE Inst. Elec. Electr. Eng.* **1962**, *50*, 2061–2070. Available online: <https://ieeexplore.ieee.org/document/4066548> (accessed on 14 August 2022). [CrossRef]
6. Lighthill, J. Artificial Intelligence: A General Survey. In *J. Art. Int. A General Survey Artificial Intelligence: A Paper Symposium*; Science Research Council: New York, NY, USA, 1972. Available online: http://www.chilton-computing.org.uk/inf/literature/reports/lighthill_report/p001.htm (accessed on 14 August 2022).
7. Towell, G.G.; Shavlik, J.W. Knowledge-based artificial neural networks. *Artif. Intell.* **1994**, *70*, 119–165. [CrossRef]
8. Towell, G.G.; Shavlik, J.W.; Noordewier, M.O. Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks. *AAAI-90 Proc.* **1990**, *2*, 861–866. Available online: <https://www.aaai.org/Papers/AAAI/1990/AAAI90-129.pdf> (accessed on 14 August 2022).
9. Towell, G.G.; Shavlik, J.W. Extracting refined rules from knowledge-based neural networks. *Mach. Learn.* **1993**, *13*, 71–101. [CrossRef]
10. Fu, L. Introduction to knowledge-based neural networks. *Knowl.-Based Syst.* **1995**, *8*, 299–300. [CrossRef]
11. Nechepurenko, L.; Voss, V.; Gritsenko, V. Comparing Knowledge-Based Reinforcement Learning to Neural Networks in a Strategy Game. In *Hybrid Artificial Intelligent Systems. HAIS 2020*; Lecture Notes in Computer Science; de la Cal, E.A., Villar Flecha, J.R., Quintián, H., Corchado, E., Eds.; Springer, Cham, Switerzland, 2020; Volume 12344. [CrossRef]
12. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [CrossRef]
13. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L. Physics-Informed machine learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440. [CrossRef]
14. Raissi, M.; Yazdani, A.; Karniadakis, G.E. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* **2020**, *367*, 1026–1030. [CrossRef]
15. Zhai, H.; Zhou, Q.; Hu, G. Predicting micro-bubble dynamics with semi-physics-informed deep learning. *AIP Adv.* **2022**, *12*, 035153. [CrossRef]
16. Zhai, H.; Sands, T. Controlling Chaos in Van Der Pol Dynamics Using Signal-Encoded Deep Learning. *Mathematics* **2022**, *10*, 453. [CrossRef]
17. Cooper, M.; Heidlauf, P.; Sands, T. Controlling Chaos—Forced van der Pol Equation. *Mathematics* **2017**, *5*, 70. [CrossRef]
18. Efheij, H.; Albagul, A. Comparison of PID and Artificial Neural Network Controller in on line of Real Time Industrial Temperature Process Control System. In Proceedings of the 2021 IEEE 1st International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering MI-STA, Tripoli, Libya, 25–27 May 2021; pp. 110–115. [CrossRef]
19. Lee, Y.-S.; Jang, D.-W. Optimization of Neural Network Based Self-Tuning PID Controllers for Second Order Mechanical Systems. *Appl. Sci.* **2021**, *11*, 8002. [CrossRef]
20. Scott, G.M.; Shavlik, J.W.; Ray, W.H. Refining PID Controllers Using Neural Networks. *Neural Comput.* **1992**, *4*, 746–757. [CrossRef]
21. Hagan, M.T.; Demuth, H.B. Neural networks for control. In Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251), San Diego, CA, USA, 2–4 June 1999; Volume 3, pp. 1642–1656. [CrossRef]
22. Nguyen, D.H.; Widrow, B. Neural networks for self-learning control systems. *IEEE Control. Syst. Mag.* **1990**, *10*, 18–23. [CrossRef]
23. Antsaklis, P.J. Neural Networks in Control Systems; Guest Editor’s Introduction. *IEEE Control. Syst. Mag.* **1990**, *10*, 3–5.
24. van der Pol, B. On “Relaxation Oscillations”. *I. Philos. Mag.* **1926**, *2*, 978–992. [CrossRef]
25. van der Pol, B.; van der Mark, J. Frequency Demultiplication. *Nature* **1927**, *120*, 363–364. [CrossRef]
26. Duke University. The van der Pol System. Available online: <https://services.math.duke.edu/education/ccp/materials/diffeq/vander/vand1.html> (accessed on 14 August 2022).
27. Partially Modified from “Phase portrait of Van-Der-Pol oscillator in TikZ”. Available online: <https://latexdraw.com/phase-portrait-of-van-der-pol-oscillator/> (accessed on 14 August 2022).
28. Schult, D. Math 329—Numerical Analysis Webpage. Available online: <http://math.colgate.edu/math329/exampleode.py> (accessed on 14 August 2022).
29. Liu, D.C.; Nocedal, J. On the limited memory BFGS method for large scale optimization. *Math. Program.* **1989**, *45*, 503–528. [CrossRef]
30. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-scale machine learning on heterogeneous systems. *arXiv* **2016**, arXiv:1603.04467.
31. Bisong, E. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Apress: Berkeley, CA, USA, 2019. [CrossRef]
32. Ahmed, H.; Ríos, H.; Salgado, I. Chapter 13—Robust Synchronization of Master Slave Chaotic Systems: A Continuous Sliding-Mode Control Approach with Experimental Study. *Recent Adv. Chaotic Syst. Synchroniz.* **2019**, 261–275. [CrossRef]
33. Hu, K.; Chung, K.-W. On the stability analysis of a pair of van der Pol oscillators with delayed self-connection, position and velocity couplings. *AIP Adv.* **2013**, *3*, 112118. [CrossRef]
34. Elfouly, M.A.; Sohaly, M.A. Van der Pol model in two-delay differential equation representation. *Sci. Rep.* **2022**, *12*, 2925. [CrossRef] [PubMed]