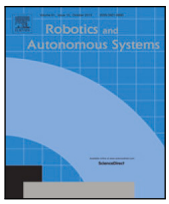
Robotics and Autonomous Systems 171 (2024) 104569

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Physics-informed Neural Network for Quadrotor Dynamical Modeling

Weibin Gu [a], Stefano Primatesta [b], Alessandro Rizzo [a,*]

[a] Department of Electronics and Telecommunications, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy
[b] Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Corso Duca degli Abruzzi, 24, Turin, 10129, Italy

ARTICLE INFO

Keywords:
Physics-informed neural network
Learning-based dynamical modeling
Interpretability
Unmanned aerial vehicle

ABSTRACT

The explosive growth of civil applications of Unmanned Aerial Vehicles (UAVs) calls for control algorithms that enable safe and trustworthy operations, especially in complex environments to facilitate real-world deployment. Although Model-Based Control (MBC) has found great applicability in the last decades, it relies heavily on the accuracy of the underlying mathematical models. Thanks to their powerful approximation capability, data-driven approaches such as Artificial Neural Networks (ANNs) have raised a great interest in UAV dynamical modeling in recent years. Despite the promising results achieved in learning either full or partial dynamics such as aerodynamic effects, prior studies merely pay little to no attention to the black-box nature of ANNs, yielding poorly interpretable learning processes and implying a lack of generalization capability due to undesirably learned spurious relationships among features and labels. While nowadays practitioners and society are far from being concerned just with performance, such risk evidently reduces the overall trustworthiness of the control system with ANNs as control-oriented models. With the goal of both accurate and interpretable neural modeling of nonlinear dynamics, we propose a novel modeling approach that utilizes Physics-Informed Neural Networks (PINN), designed to seamlessly embed physical laws into Deep Neural Networks (DNN). To further enhance model interpretability, we adopt the Covariance Confidence Ellipse (CCE) as post-hoc visualization to help understand the model behavior. We also set up a visual and physical simulator based on AirSim with custom implementation on quadrotor ground effect, which is made publicly available to facilitate further experiments for the community. An extensive simulation campaign validates our approach on data sets including aerodynamics and periodic wind, highlighting advantages regarding the accuracy and physical consistency, pointing to further insight on the development of learning-based control-oriented models for quadrotors, and facilitating flight controller design with performance guarantees.

1. Introduction

The 21st century has witnessed an explosive growth of civil applications of Unmanned Aerial Vehicles (UAVs), ranging from the inspection of industrial infrastructures such as power lines [1] and wind turbines [2], to operations in human-interactive environments such as delivery of goods [3], to name a few [4]. Such a scenario calls for control algorithms that not only provide sufficient tracking performance, but also enable *safe* and *trustworthy* operations, especially in complex and populated environments, to facilitate real-world deployment as well as to avoid injury and property damage by all means.

Model-Based Control (MBC) techniques have found great applicability in the last decades thanks to the underlying analytical formulation of system dynamics in terms of differential equations, which typically informs the design of *performance-guaranteed* control techniques [5]. However, a remarkable drawback of MBC techniques is their heavy reliance on the accuracy of the mathematical model of the system under control [6]. *Uncertainties* and *disturbances*, such as parametric uncertainty and unmodeled dynamics, are ubiquitous in real-world flight, which may set back the derivation of such control-oriented models, with a clear adverse impact on the control performance. A typical example is the loss of system stability, which, for UAVs, results in dangerous and uncontrollable deviations from the planned path, eventually incurring accidents [7].

To address the modeling issues induced by uncertainties and disturbances, many efforts in the control community have been put forward in robust and adaptive control theory [6]. For example, adaptive backstepping controllers were designed to account for the changes in mass and inertia matrix of UAV [8] and in payload [3]; another adaptive sliding backstepping control scheme was proposed to guarantee attitude tracking under unmodeled dynamics [9]. Nonetheless, most of the resulting controllers are usually mathematically complex and the design very often leans toward conservative approaches, tuned

* Corresponding author.
E-mail addresses: weibin.gu@polito.it (W. Gu), stefano.primatesta@polito.it (S. Primatesta), alessandro.rizzo@polito.it (A. Rizzo).

https://doi.org/10.1016/j.robot.2023.104569
Received 29 May 2023; Accepted 24 October 2023
Available online 2 November 2023

on the demanding, yet unlikely, worst-case scenario, thus limiting the possibilities of guaranteeing desired performance [6].

On the other side of the spectrum, data-driven approaches have been pursued with the aim to learn the real system dynamics in its entire complexity from empirical data sets, including uncertainties and disturbances, which could be subsequently used for MBC design. Artificial Neural Networks (ANNs), as *universal approximators* with powerful learning capability, have probably raised the most interest for UAV dynamical modeling due to the advances in computer hardware in recent years [10–13]. As they started to be applied to the growing UAV industry, *trustworthiness* has become an exigent demand, which by definition [14], requires human understanding and trust, especially for new techniques. Due to the *black-box* (a.k.a., model-agnostic) nature of ANNs, however, the performance metrics mainly rely on the assessment of input–output numerical deviations of the available data. Hence, the outcome of the learning process is typically *poorly interpretable*, often implying a lack of generalization capability outside the training data set due to undesirably learned spurious relationships among features and labels. This issue becomes particularly harmful when ANNs are used as a control-oriented model to design control actions, since such systems respond to external stimuli with behaviors that are maybe effective yet inexplicable. While nowadays practitioners and society are far from being concerned just by performance, such risk evidently reduces the overall trustworthiness of the control system [14].

To meet the expectations of both *accurate* and *interpretable* neural modeling, here we propose a novel approach for dynamical modeling of quadrotors, inspired by *Physics-Informed Machine Learning (PIML)* [15], an emerging machine learning paradigm that aims at achieving better generalization capability by incorporating a-priori system knowledge into the learning process. Such paradigm has been successfully applied for modeling in many scientific and engineering disciplines such as the acoustic field of a quadrotor [16–18], robotic manipulators [19–21], lake temperature [22], pandemic spread [23], just to list a few. In our context, we seamlessly embed the law of conservation of momentum into the training of a Deep Neural Network (DNN) in the form of *soft constraints* (or *learning bias*) to model fast, high-dimensional, and highly nonlinear dynamics. Comparison and ablation studies have been carried out over multiple seeds in a visual and physical simulator that we customized on top of AirSim [24] on several data sets including aerodynamics such as drag, ground effect, and periodic wind. It has been consistently shown that our proposed Physics-Informed Neural Network (PINN) outperforms both linearized mathematical models and purely black-box approaches such as vanilla DNN [11] in terms of test error (generalizability on unseen data) and exhibits better learning capability of underlying relationships than vanilla DNN by means of Covariance Confidence Ellipse (CCE) [25], a *post-hoc* model interpretability technique introduced to reveal *physical consistency* of the learned model.

The remainder of this paper is organized as follows. We start with providing an overview on the prior studies on learning-based dynamical modeling of UAVs and the development of PIML in learning system dynamics in Section 2, followed by the highlight on our main contributions that distinguish our work from the state of the art. In Section 3, we summarize the well-developed mathematical model of quadrotors with notations and formulate the problem that we aim to solve within this paper. Section 4 elaborates the main idea of our proposed PINN for quadrotor dynamical modeling accompanied by all the other details of our network design. We then introduce the visual and physical simulator that we set up and customized for simulating advanced aerodynamics such as ground effect and collecting flight data for network training in Section 5, where main results from multiple seed campaigns as well as discussion are also given to reveal the advantages of our proposed approach. Finally, Section 6 draws our conclusions, provides main takeaways, and points out some future research directions.

2. Related work and contributions

Recent efforts mostly related to our work are contributed by two distinct communities, i.e., *UAV control* and *machine learning*, yet with different research focus as outlined below.

2.1. Learning-based dynamical modeling of UAV

Prior studies in the UAV control community have widely proved the effectiveness of deep learning techniques for dynamical modeling of UAV thanks to their great potential in capturing complex features hidden in the data. For example, DNN was successfully adopted to learn the dynamics of helicopter [10] and quadrotor [11] from real flight data. Besides direct modeling of the full dynamics, it has been recently studied that utilizing DNN to learn the residual forces and/or torques due to advanced aerodynamics is feasible such as quadrotor ground effect [12] and aerodynamic interactions between multirotors [13]. Despite the satisfactory results achieved in these studies, vanilla DNN is commonly selected as the modeling architecture, which is notorious for its purely black-box characteristics, thereby susceptible to learn physically inconsistent relationships and leading to degraded generalization capability [15].

In regard to this issue, only a few attempts have been made so far to seek ways to make the learned model more interpretable. In [26], a hybridization of analytical and empirical techniques was proposed for linear velocity estimation of a quadrotor, where the data-driven model was trained by a custom loss function, i.e., the average of the sum of Mean Squared Errors (MSEs) of prediction error, step response, and the gain of response given a sinusoidal input signal. Similar idea of hybridization was adopted in [27] where quadrotor model was constructed by the combination of Blade-Element-Momentum (BEM) theory with a neural network compensating the residual dynamics using real flight data of agile maneuvers. However, the structure and the training process of the employed network used in these learning-based models still remain black-box. Very recently, Domain Adversarially Invariant Meta-Learning (DAIML) was proposed to learn offline the shared representation of winds and then a composite adaptation law was designed to update online the wind-specific linear coefficients of this basis function [28]. Clustered linear coefficients with similar wind speed can be visualized a posteriori which implies DAIML succeeds in learning the basis function shared by all wind conditions, thus proving the interpretability of the approach. Nonetheless, this approach requires a very representative data collection that covers different wind speeds, which could be very laborious and time-consuming.

2.2. Learning dynamics from trajectory data using PINN

With the recent success of solving Partial Differential Equations (PDEs) by using PINN [29], research on learning dynamics from trajectory data by leveraging physical insights began to flourish among machine learning scientists. Here we broadly classify these works into *structured* and *unstructured* learning.

Lagrangian neural networks [19–21] and Hamiltonian neural networks [30,31] are two typical frameworks belonging to structured learning, which use Lagrangian and Hamiltonian mechanics to inform the structure of neural Ordinary Differential Equation (ODE) of the system. The major advantage of these structured learning techniques is that energy conservation can be guaranteed if the system does not include non-conservative forces such as friction. Despite the fact that this could be true, especially for the dynamical modeling of robotic manipulators, aerodynamic forces can make significant impact during a quadrotor's flight, thereby making these frameworks challenging to be applied in our proposed scenario.

On the other side, unstructured learning aims to incorporate a-priori domain knowledge into the loss function as learning bias to achieve more physically consistent predictions. In some literature, this

is also known as Theory-Guided Data Science (TGDS) [32]. The crux of unstructured learning is the embedding of knowledge from the beginning [14] and the assessment of its compliance with the knowledge eventually learned by the model. Continuous investigations in recent years in both scientific and engineering disciplines have also shown the effectiveness of this line of research such as acoustic field modeling of a quadrotor [16–18], lake temperature modeling [22] and pandemic spread prediction [23].

2.3. Main contributions

This work falls under the category of unstructured learning, where a PINN is designed for modeling the full dynamics of a quadrotor. The reason for not doing residual modeling is that advanced aerodynamics are in general too complex to be analytically modeled, due to their highly nonlinear characteristics. Thus, finding an appropriate prior domain knowledge to be incorporated into PINN for modeling these aerodynamic forces becomes extremely challenging. As an example, despite the fact that recent studies on quadrotor ground effect have revealed some critical variables based on the analyses on empirical data [33,34], there still does not exist a widely acknowledged mathematical model for all possible quadrotor configurations and flight maneuvers. To bypass this obstacle, we exploit the well-known law of conservation of momentum in the full dynamical modeling, which helps inform the network even when non-conservative forces exist. Our main contributions are summarized as follows:

- We propose a PINN for full dynamical quadrotor modeling informed by the law of conservation of momentum, which is embedded into the training loss function as learning bias via a local monotonicity function.
- We implemente a parametric model of quadrotor ground effect derived from empirical data [33] as a custom class[1] to the visual and physical simulator, AirSim [24], for flight data collection, also provided with high-level C++ and Python user Application Programming Interfaces (APIs) to facilitate further research in the UAV community.
- We carry out extensive comparison and ablation studies over multiple seeds to evaluate the advantages of our proposed PINN over linearized mathematical model and vanilla DNN. In particular, we adopt CCE as post-hoc model interpretability technique to visualize physical consistency of the learned model.

3. Mathematical model of quadrotor and problem formulation

3.1. Quadrotor dynamics

The kino-dynamic model of a quadrotor can be formulated through Eqs. (1) [35]. There, we consider North-East-Down (NED) inertial and Front-Right-Down (FRD) body-fixed coordinates as the adopted reference frames indicated by superscripts $\mathcal{I}$ and $\mathcal{B}$, respectively, and a generic quadrotor configuration, as depicted in Fig. 4(c). We further denote the position of the quadrotor by ${}^{\mathcal{I}}\mathbf{p} = (p_x, p_y, p_z)^{\mathrm{T}} \in \mathbb{R}^3$, linear velocity by ${}^{\mathcal{I}}\mathbf{v} = (v_x, v_y, v_z)^{\mathrm{T}} \in \mathbb{R}^3$, attitude rotation matrix from the body frame to the inertial frame by $\mathbf{R}_{\mathcal{B}}^{\mathcal{I}} \in \mathrm{SO}(3)$ expressed in Euler angles $\boldsymbol{\eta} = (\phi, \theta, \psi)^{\mathrm{T}} \in \mathbb{R}^3$, body-fixed angular rate by ${}^{\mathcal{B}}\boldsymbol{\omega} = (p, q, r)^{\mathrm{T}} \in \mathbb{R}^3$, and mass and inertia matrices as $m \in \mathbb{R}$ and $\mathbf{J} = \mathrm{diag}(J_x, J_y, J_z) \in \mathbb{R}^3$, respectively. Moreover, ${}^{\mathcal{B}}\mathbf{f}_{\mathrm{u}} = (0, 0, T)^{\mathrm{T}} \in \mathbb{R}^3$ and ${}^{\mathcal{B}}\boldsymbol{\tau}_{\mathrm{u}} = (\tau_{u,x}, \tau_{u,y}, \tau_{u,z})^{\mathrm{T}} \in \mathbb{R}^3$ are the total thrust and body torque generated by the four rotors, ${}^{\mathcal{B}}\mathbf{f}_{\mathrm{a}}, {}^{\mathcal{B}}\boldsymbol{\tau}_{\mathrm{a}} \in \mathbb{R}^3$ are the aerodynamic forces and torques such as drag and ground effect, and $\mathbf{g} = (0, 0, g)^{\mathrm{T}} \in \mathbb{R}^3$ is the gravity vector with gravitational acceleration constant g. Lastly, $\boldsymbol{\Phi}$ is the transformation matrix that converts body angular rate to the variation of Euler angles.

$$
\begin{aligned}
{}^{\mathcal{I}}\dot{\mathbf{p}} &= {}^{\mathcal{I}}\mathbf{v} \\
m{}^{\mathcal{B}}\dot{\mathbf{v}} &= -m{}^{\mathcal{B}}\boldsymbol{\omega} \times {}^{\mathcal{B}}\mathbf{v} + \mathbf{R}_{\mathcal{B}}^{\mathcal{I}} m{}^{\mathcal{I}}\mathbf{g} + {}^{\mathcal{B}}\mathbf{f}_u + {}^{\mathcal{B}}\mathbf{f}_a \\
\dot{\boldsymbol{\eta}} &= \boldsymbol{\Phi}{}^{\mathcal{B}}\boldsymbol{\omega} \\
\mathbf{J}{}^{\mathcal{B}}\dot{\boldsymbol{\omega}} &= \mathbf{J}{}^{\mathcal{B}}\boldsymbol{\omega} \times {}^{\mathcal{B}}\boldsymbol{\omega} + {}^{\mathcal{B}}\boldsymbol{\tau}_u + {}^{\mathcal{B}}\boldsymbol{\tau}_a
\end{aligned}
\tag{1}
$$

3.2. Rotor model

The overall quadrotor model is completed by taking into account the rotor model, which associates the generated thrust T and torque ${}^{\mathcal{B}}\boldsymbol{\tau}_u$ with the rotor angular speeds n_i $(i = 1, 2, 3, 4)$ in revolutions per second as given in Eq. (2), where c_T is the dimensional thrust coefficient, c_Q is the dimensional moment coefficient, and l is the moment arm, i.e., the distance from the rotor axis to the principal axis of the quadrotor.

$$
\begin{bmatrix} T \\ \tau_{u,x} \\ \tau_{u,y} \\ \tau_{u,z} \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ -c_T l & c_T l & c_T l & -c_T l \\ c_T l & -c_T l & c_T l & -c_T l \\ c_Q & c_Q & -c_Q & -c_Q \end{bmatrix} \begin{bmatrix} n_1^2 \\ n_2^2 \\ n_3^2 \\ n_4^2 \end{bmatrix}
\tag{2}
$$

3.3. Linearized model

To facilitate the design of the controller, a simplified model is commonly adopted, which can be derived from the linearization of Eqs. (1) around a stable hovering condition, under the assumption that aerodynamic forces and torques are negligible. This yields the linearized equations of translational and rotational dynamics Eq. (3). While such a linearized model provides a good approximation of the quadrotor dynamics close to the hovering condition, conditions far from the equilibrium such as agile maneuvers or stringent control performance call for the usage of a more accurate, if not complete, dynamical model that includes nonlinearities and aerodynamics factors [7].

$$
\begin{aligned}
\dot{v}_x &= -\frac{1}{m}\theta T, \quad \dot{v}_y = \frac{1}{m}\phi T, \quad \dot{v}_z = -\frac{1}{m}T + g \\
\dot{p} &= \frac{1}{J_x}\tau_{u,x}, \quad \dot{q} = \frac{1}{J_y}\tau_{u,y}, \quad \dot{r} = \frac{1}{J_z}\tau_{u,z}
\end{aligned}
\tag{3}
$$

3.4. Problem formulation

It is well acknowledged that aerodynamic forces and torques caused by the interaction effects among the propellers such as drag and ground effect become significant when the quadrotor is commanded to execute more agile maneuvers. In this work, instead of learning these residual dynamics using DNNs, we aim to approximate the full dynamical model including these unknowns by a PINN, which is informed by a known physical law during network training to pursue better generalization capability and enhance interpretability. The PINN takes quadrotor states and accelerations as input features and predicts the motor speeds, thereby serving as an inverse dynamical model that could be applied with feedback linearization control technique.

4. Physics-Informed Neural Network

In this section, we elaborate the proposed PINN for modeling the inverse dynamics of the quadrotor. Following unstructured learning strategy, the main objective is to find appropriate domain knowledge and incorporate it into network training so that the resulting network can make more physically consistent predictions. This can also be regarded as constructing a custom physics-informed loss function that guides the network, which can take any arbitrary structure such as feedforward neural network or Recurrent Neural Network (RNN), to learn the inherent relationship among data. In the sequel, we first detail the network structure that we adopt throughout this work, followed by reasoning and explanation of how we incorporate physics into network

[1] Our custom implementation of ground effect in AirSim: https://gitlab.com/PoliToComplexSystemLab/AirSim-GE

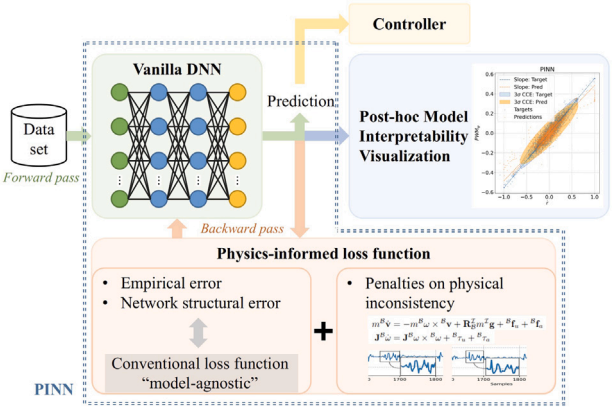


Fig. 1. Overall structure of PINN. PINN is indeed a vanilla DNN trained with a physics-informed loss function, in which prior domain knowledge is embedded as opposed to the model-agnostic conventional loss function. In addition to the prediction error, the output of the network will also be evaluated by means of post-hoc visualization to see how well the network actually learns the prior knowledge and can be further used in the controller design.

training by devising a physics-informed loss function. We also shed light on a post-hoc model interpretability visualization for verifying network capability of extracting underlying knowledge and a practical trick of adjusting regularization to achieve better training performance for our application. The overall structure of the proposed PINN is depicted in Fig. 1.

4.1. Network structure

The proposed PINN, which approximates the inverse dynamics of the quadrotor, can be expressed as $\hat{Y} = \text{PINN}(X;\Theta)$, where X denotes the network input in the form of a vector consisting of quadrotor states and accelerations, $\hat{Y}$ denotes the network predictions on control inputs, and Θ denotes the trainable network parameters. Since yaw angle may suffer from abrupt changes from 0 to 2π, sine and cosine operators are used for embedding to ensure the continuity of the signal. Besides, we use the dimensionless form of Pulse Width Modulation (PWM) of rotors as control inputs due to the signal property from the visual and physical simulator that we set up for data collection as will be discussed later. Therefore, the input and output of the network can be formulated as follows:

- $X := ({}^{\mathcal{B}}\dot{\mathbf{v}}^{\mathrm{T}}, {}^{\mathcal{B}}\dot{\boldsymbol{\omega}}^{\mathrm{T}}, {}^{\mathcal{B}}\mathbf{v}^{\mathrm{T}}, {}^{\mathcal{B}}\boldsymbol{\omega}^{\mathrm{T}}, \phi, \theta, \sin\psi, \cos\psi)^{\mathrm{T}} \in \mathbb{R}^{15}$
- $\hat{Y} := (\hat{u}_1, \hat{u}_2, \hat{u}_3, \hat{u}_4)^{\mathrm{T}} \in \mathbb{R}^4$, where u_i is the PWM signal of each rotor

We adopt DNN as the structure of our network and hence, our PINN can be re-written as Eq. (4), where $\{\mathbf{W}^{[1]}, \mathbf{b}^{[1]}\}$ is the pair of network parameters of the first layer in Θ, $g(\cdot) = \text{ReLU}(\cdot)$ is the activation function, and $f(\boldsymbol{\beta}) = \mathbf{W}^{[i]}\boldsymbol{\beta} + \mathbf{b}^{[i]}$, with $i \in [1, N_l]$. Note that if the model in Eq. (4) is trained with conventional loss function, i.e., MSE between Y and $\hat{Y}$, then it becomes the vanilla DNN (e.g., see [11]). However, we elaborate our custom physics-informed loss function in the following, which is specifically designed with the aim of guiding the network to learn the inherent relationship between input–output data so as to satisfactorily generalize on unseen data sets.The network has therefore the form

$$\text{PINN}(X,\Theta) := f \circ \underbrace{g \circ \cdots \circ f \circ g}_{N_l \text{ hidden layers}} (\mathbf{W}^{[1]}X + \mathbf{b}^{[1]}) \tag{4}$$

4.2. Incorporation of physics as learning bias

As mentioned in Section 2.3, the fact that no widely acknowledged mathematical representations or even empirical domain expertise of advanced aerodynamics exist, makes incorporating this domain knowledge extremely challenging. Therefore, we pivot to leverage an ubiquitous physical law to guide the training of the network, which, for quadrotors, is the law of conservation of momentum. Such a law inherently governs the dynamical equations and can be considered as a fundamental property of the system. As such, the problem then boils down to incorporating this physical law into network training through suitable loss terms.

The embedding of the law of conservation of momentum is motivated by the correlation of accelerations and control inputs. Taking our collected data (of agile maneuvers with drag) as an example, as shown in Fig. 2, it can be easily evaluated the Pearson Correlation Coefficient (PCC) of the derivative of roll angular rate $\dot{p}$ and the PWM signals used to generate such acceleration, which has a numerical value of 0.67. This means that these two quantities have a strong correlation and graphically, these data series share a *similar pattern* to a great extent (see Figs. 2 (a)&(b)). Such strong correlation also holds positively for pitch and yaw motion (see Figs. 2 (e)&(f), (i)&(j)). For better visualization, we plot the sorted samples with CCE [25], which represents the enclosure of 98.9% of data, as shown in Figs. 2 (d)&(h)&(i). The positive correlation can then be implied by the slope (or more precisely, the rotation angle) of the CCEs. From the perspectives of mathematical formulation, it can be observed that $\dot{\omega}$ and ${}^{\mathcal{B}}\tau_u$ are collinear when nonlinearities and aerodynamics are marginal as shown in Eq. (3).

Based on the above empirical observations, we devised a local monotonicity loss term to embed the law of conservation of momentum into the loss function for network training. The local monotonicity loss term penalizes the violation of two data series in terms of inconsistent patterns. More specifically, if the first data series increases whereas the second data series in the meanwhile decreases, then the network will be penalized during the training by adding this local monotonicity loss term into the final loss function. The detailed implementation of the local monotonicity loss term is given in Algorithm 1, where we use $\tanh(\cdot)$, a differentiable approximation of the sign function, for examining if two data series share the same pattern. Moreover, we impose the local monotonicity loss to the quantities of all the three degrees of rotation, which results in the final physics-informed loss function for network training as given in Eq. (5) where λ_{LM} is the hyperparameter and $\mathcal{L}_{\text{MSE}}$ is the (conventional) MSE between targets and predictions.

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \sum_i \lambda^i_{\text{LM}}\mathcal{L}^i_{\text{LM}}, \quad i = \{\phi, \theta, \psi\} \tag{5}$$

A few more comments on the proposed physics-informed loss function are in order. First, one may notice that we do not directly embed PCC into the composition of the final loss function. This is because PCC is dependent on assumptions on the data distribution and it may yield poor training performance, whereas it is an effective metric to assess the correlation between variables, once the data vectors are available. Notably, it may happen that two data series with different patterns might have the same value of PCC.[2] Hence, using PCC for training would steer the network parameters toward similar values even in presence of different patterns in input, clearly yielding ill-posed optimization problems. Second, the observation of sharing the same

[2] An explanatory example: Let the first pair of data series be $x_1 = [1,2,3,4,5]$, $y_1 = [1,-2,-3,4,5]$ and the second pair of data series be $x_2 = [1,2,3,4,5]$, $y_2 = [1,2,3,1.9455,2.5]$. It is straightforward to see that the increasing/decreasing pattern between x_i and y_i where $i = 1,2$ is significantly different. However, PCC cannot tell the difference (both two pairs of data series have PCC of approximately 0.626) whereas the proposed local monotonicity loss can do it (1.0 for $i = 1$ and 0.5 for $i = 2$).

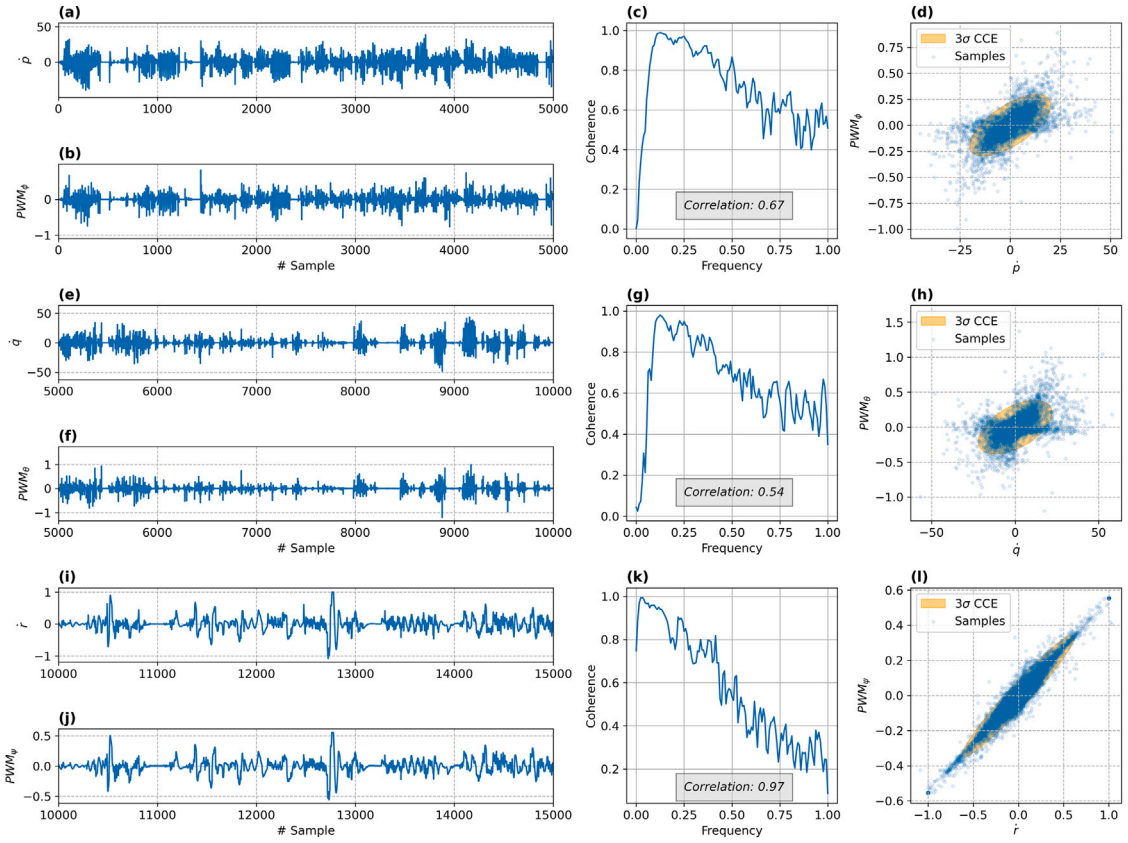


Fig. 2. Correlation within training data series. (a), (b), (e), (f), (i), (j): Data series of the derivative of angular rate and the corresponding PWM signals to generate such acceleration. **(c), (g), (k):** Coherence plots between the data series for roll, pitch, and yaw motion, where the calculated PCCs are also indicated in gray boxes. **(d), (h), (i):** CCE plots between the data series for roll, pitch, and yaw motion, where samples are depicted by blue dots and 3σ CCE is depicted by orange ellipses. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

pattern does not always hold, and hence neither does the devised local monotonicity loss. This is because our proposed embedding (i.e., local monotonicity term) is also an approximation rather than rigorous mathematical equation. However, this could happen only when the system dynamics are dominated by the highly nonlinear aerodynamic terms. The solution to this is to include the local monotonicity loss as an auxiliary learning bias, i.e., a soft constraint (hence, can be violated) that helps improve the network learning through the hyperparameter λ_{LM}, which is scheduled by cyclical annealing method [36]. Third, compared to the conventional MSE loss function used for training the vanilla DNN, the additional computational complexity of our physics-informed loss function mainly comes from the inclusion of the local monotonicity loss (i.e., Algorithm 1), which is composed of $\tanh(\cdot)$ and other basic arithmetic operators. We will show in Section 5 that such additional computational complexity is generally acceptable for non-time-critical situations by providing empirical evidence.

4.3. Cyclical annealing scheduler

As mentioned in the previous section, the two loss terms in Eq. (5) contradict each other when aerodynamics prevail over the body wrench generated by the four rotors. Hence, the local monotonicity loss fails to guide correctly the network to learn the real aerodynamic effect. To address this issue, we adopt a practical trick, namely cyclical annealing scheduler, which was first introduced to machine learning field to mitigate Kullback–Leibler (KL) vanishing problem of Variational Autoencoder (VAE) [36]. Similar to our physics-informed loss function, VAE has a loss function consisting of MSE (i.e., reconstruction error) and KL regularizer. By scheduling the hyperparameter of KL regularizer in a cyclical fashion, it was empirically found that the network can leverage latent codes learned in the previous cycle as warm re-starts, thereby improving the performance progressively.

In our case, we apply a similar idea but with inverse scheduling as given in Eq. (6), where k is the current epoch index, T (with a slight abuse of notation) is the maximum number of epochs, $\lambda_{\max}$ is the maximum value to which λ_{LM} will be annealed, M is the number of cycles, and R is the proportion used to maintain $\lambda_{\max}$. As such, the training process is split into M cycles, each starting from $\lambda_{\max}$ at which the network tends to abide by local monotonicity, to 0 when our physics-informed loss function recovers the conventional loss function and attempts to simply minimize the MSE with its all effort (see Fig. 3). This scheduling strategy is shown to be effective in achieving better training performance in Section 5. Such a strategy can be mathematically summarized as

$$\lambda_{\text{LM}} = \begin{cases} \frac{1-\beta}{1-R}\lambda_{\max} & \text{if } \beta > R \\ \lambda_{\max} & \text{if } \beta \le R \end{cases} \qquad \beta = \frac{\text{mod}(k, \lceil T/M \rceil)}{\lceil T/M \rceil} \tag{6}$$

4.4. Post-hoc model interpretability visualization

To design trustworthy artificial intelligence, besides developing transparent models such as our proposed PINN, a parallel way is to apply post-hoc model interpretability techniques including explanations by example, text explanations, feature relevance, and so forth [14]. Here, we apply post-hoc visualization to further improve the interpretability of the learned model. Different from [28], where a dimensionality reduction technique was adopted to cluster different wind

Algorithm 1: Calculation of physics-informed loss function

Input: Training data $\mathcal{D}$, network parameter Θ
Parameter: Batch size N_b, hyperparameter λ_{LM}
Output: Physics-informed loss $\mathcal{L}$

Function `LocalMonotonicityLoss(m, n):`
  LM_m, $\text{LM}_n \leftarrow \tanh(m[2, N_b] - m[1, N_b - 1])$, $\tanh(n[2, N_b] - n[1, N_b - 1])$
  $\text{loss} \leftarrow \frac{1}{N_b}(1 - \text{LM}_m \times \text{LM}_n)$ `// element-wise multiplication`
  return loss
$X_{batch}, Y_{batch} \leftarrow \text{DataLoader}(D, N_b)$ `// X: features, Y: labels`
$\hat{Y}_{batch} \leftarrow \text{PINN}(X_{batch}, \Theta)$ `// forward propagation`
$\mathcal{L}_{\text{MSE}} \leftarrow \frac{1}{N_b}(\hat{Y}_{batch} - Y_{batch})^2$ `// (conventional) MSE`
$\mathcal{L}_{\text{LM}} \leftarrow \text{LocalMonotonicityLoss}(\hat{Y}_{batch}, X_{batch})$ `// local monotonicity loss`
$\mathcal{L} \leftarrow \mathcal{L}_{\text{MSE}} + \lambda_{\text{LM}}\mathcal{L}_{\text{LM}}$
return $\mathcal{L}$

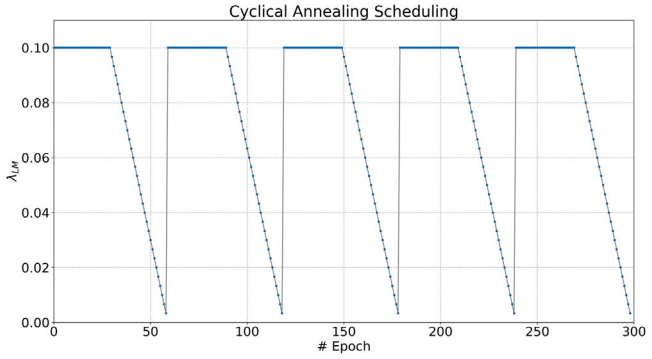


Fig. 3. Illustration of cyclical annealing scheduler λ_{LM} in Eq. (6) with parameters: $(M, R, \lambda_{\max}, T) = (5, 0.5, 0.1, 300)$.

conditions, we use CCE as already mentioned in the previous sections due to the fact that it can roughly reflect PCC (by means of rotation angle of CCE) and MSE (by means of height and width of CCE). Such visualization can be very helpful in understanding the model behavior in either training or deployment, especially for users who are not familiar with machine learning.

5. Results and discussion

5.1. Visual and physical simulator

To facilitate data collection, a simulator based on the visual and physical simulator AirSim [24], Unreal Engine 4 (UE4), PX4, and QGroundControl (see Figs. 4 (a) and (b)), has been set up in our work. Compared to other open-source simulator alternatives such as Gazebo,[3] X-Plane[4] and asbQuadcopter by MathWorks,[5] AirSim provides a more complex and realistic environment by building on top of a physics engine with advanced rendering techniques made by UE4 and featuring various sensors and aerodynamics features such as drag forces. Thanks to its excellent extensibility, it is also possible to accommodate various types of hardware platforms and software protocols and to customize personalized APIs and functionalities using C++ and Python.

In our experiments, besides data collection coding, we implemented a custom C++ class on top of AirSim's source code for simulating the quadrotor ground effect to further enhance the fidelity of our simulator. Specifically, two ground effect models have been included, i.e., a simple Cheeseman–Bennett model [37] and a parametric model derived from polynomial fitting using empirical data [33] (see Fig. 5). We also provide user-level APIs in both C++ and Python, namely "simSetGroundEffect", to facilitate the research of other researchers in the UAV community. Our implementation complements the advanced aerodynamics already available in AirSim such as drag and helps users safely collect representative flight data under ground effect by flying the quadrotor very close to the ground in the simulator. More design details and usage instructions of our implementation of ground effect are referred to the GitLab repository with a self-explanatory README file therein: https://gitlab.com/PoliToComplexSystemLab/AirSim-GE.

Table 1
Quadrotor specifications.

Parameter	Symbol	Value
Mass	m	1.5 kg
Inertia	J_x	1.469e−2 kgm^2
	J_y	1.686e−2 kgm^2
	J_z	3.093e−2 kgm^2
Thrust constant	C_T	1.099e−1
Power constant	C_P	4.016e−2
Propeller diameter	D	0.2286 m
Propeller spacing	$2l$	0.690 m

5.2. Network training

Thanks to the visual and physical simulator that we have set up in Section 5.1, training data set can be easily and safely collected through manual flight of the quadrotor with specifications as given in Table 1 (note that thrust and power constants reported therein are dimensionless), using joystick or gamepad for different maneuvers with aerodynamics and other uncertainties included. In total, we collected three data sets for our experiments:

- D_1: agile maneuver with maximum linear speed up to 8 m/s and drag force, yielding approximately 15k data samples (see Fig. 4(d))
- D_2: near-ground flight with both drag force and ground effect [33], yielding approximately 6k data samples (see Fig. 5)
- D_3: agile maneuver with both drag force and periodic wind, yielding approximately 5k data samples (see Fig. 6)

All three data sets include physical quantities of the quadrotor needed for network training and ablation studies with a fixed sampling time of 0.05 sec.

A trial-and-error procedure led to the selection of 10 hidden layers, each with 25 hidden neurons, to carry out the training of neural networks with satisfactory performance. Before training, we initialized

[3] Gazebo: https://staging.gazebosim.org/home
[4] X-Plane: https://www.x-plane.com/
[5] MathWorks asbQuadcopter project: https://it.mathworks.com/help/aeroblks/quadcopter-project.html

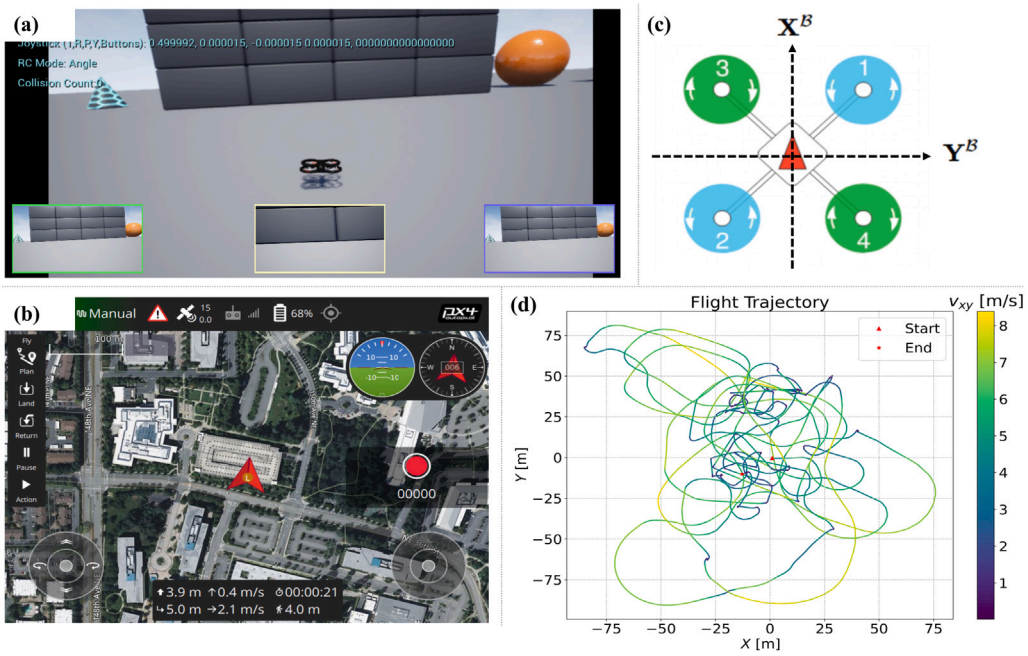


Fig. 4. Data collection process. (a) Manual flight of quadrotor in the visual and physical simulator. **(b)** GUI of QGroundControl. **(c)** Configuration of the employed quadrotor superimposed with body-fixed reference frame. **(d)** Flight trajectory with colorbar indicating the magnitude of linear velocity.

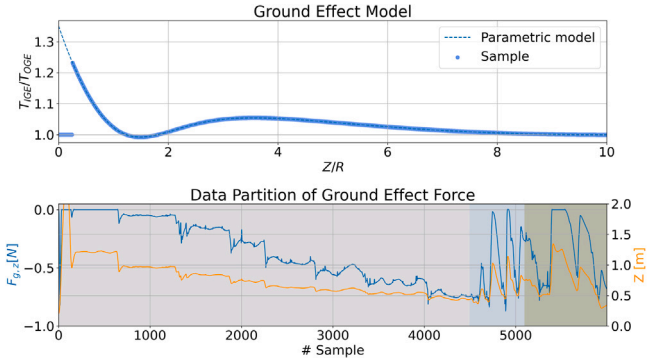


Fig. 5. Data set of simulated flight corrupted with ground effect. Top: Parametric model of quadrotor ground effect [33] (x-axis represents the ratio between altitude above the ground, Z, and propeller radius, R, and y-axis represents the ratio between the lift experienced while hovering IGE and OGE) and data samples collected from our implementation in the simulator. **Bottom:** Collected flight data for training: blue curve indicates ground effect force along body z-axis, $F_{g,z}$, orange curve indicates the altitude above the ground of the quadrotor, and shaded areas represent data partition into training/validation/test data set. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

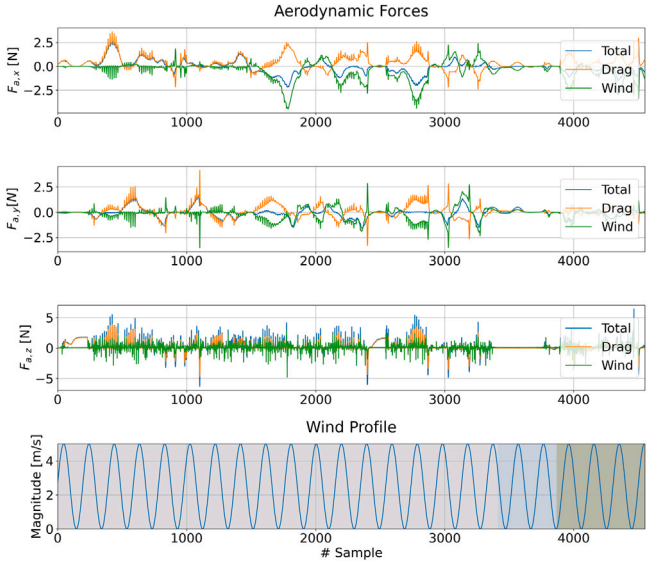


Fig. 6. Data set of simulated flight corrupted with periodic wind. The top three plots illustrate the composition of aerodynamic forces including drag and wind. The bottom plot shows the profile of periodic wind in forward direction with speed $v_{wind} = 2.5\sin(\frac{\pi t}{5}) + 2.5$, superimposed with shaded areas representing data partition into training/validation/test data set.

all the network parameters (i.e., Θ) using Xavier initialization method and picked the size for each batch of 64 data samples. Adam optimizer with weight decay (i.e., an alternative of network structural error) was used for training the network to avoid the overfitting problem.

5.3. Model comparison and ablation study

To demonstrate the effectiveness of the proposed PINN, we performed extensive comparisons on a bench of models which can be categorized into: (i) linearized mathematical model, (ii) purely black-box model (i.e., vanilla DNN similar to [11]), and (iii) PINN model (with same structure settings as the vanilla one). Moreover, ablation study was also carried out to investigate the influence of using Batch Normalization (BN) and cyclical annealing scheduler along with our PINN. A brief overview of all the models considered in our comparison and ablation studies is reported in Table 2.

All the three test data sets were used for evaluation, taking into account drag, ground effect, and periodic wind. In our assessment trials, all the considered models were evaluated in a single setting with one selected test data set for 20 times, of which results were averaged over multiple seeds and reported in Table 3. Noted that not only the prediction error (ΔY) is considered as performance metrics in our evaluation, but also the difference of slope (or rotation angle) between prediction and label CCEs, denoted by Δm, is treated as an

Table 2
Models for comparison and ablation studies. To reveal the effectiveness of our proposed approach, our particular interest is paid to: (i) the structure of the model, (ii) whether BN is applied or not, (iii) how the regularization hyperparameters are set during the training, and (iv) on which data set is trained the model. Bullets (•) symbolizes the highlighted features present within the model.

Model	Structure			BN		Hyperparameter		Data set			
ID	Linearized model	Vanilla DNN	PINN	w/o	w/	Constant	Cyclical annealing	D_1	D_2	D_3	Reduced data set
1	•	∘	∘	n/a	n/a	n/a	n/a	•	∘	∘	∘
2	∘	•	∘	•	∘	n/a	n/a	•	∘	∘	∘
3	∘	∘	•	•	∘	•	∘	•	∘	∘	∘
4	∘	∘	•	•	∘	∘	•	•	∘	∘	∘
5	∘	•	∘	∘	•	n/a	n/a	•	∘	∘	∘
6	∘	∘	•	∘	•	∘	•	•	∘	∘	∘
7	∘	•	∘	∘	•	n/a	n/a	•	∘	∘	•
8	∘	∘	•	∘	•	∘	•	•	∘	∘	•
9	∘	•	∘	∘	•	n/a	n/a	•	∘	∘	•
10	∘	∘	•	∘	•	∘	•	•	∘	∘	•
11	∘	•	∘	∘	•	n/a	n/a	∘	•	∘	∘
12	∘	∘	•	∘	•	∘	•	∘	•	∘	∘
13	∘	•	∘	∘	•	n/a	n/a	∘	∘	•	∘
14	∘	∘	•	∘	•	∘	•	∘	∘	•	∘

Table 3
Comparison results. ΔY denotes the absolute value of prediction error between prediction $\hat{Y}$ and label Y. Δm_i denotes the absolute difference of slope (or rotation angle) between prediction and label CCEs where $i = \phi, \theta, \psi$ denotes the three degree of rotation. $\bar{(\cdot)}$ denotes averaging over multiple seeds and $\sigma(\cdot)$ denotes the corresponding standard deviation. Superscript † and ‡ denote linearized mathematical model (Eq. (3)) and vanilla DNN, respectively. The results of comparison and ablation studies are reported in group which are splitted by horizontal lines with the best performance indicated in bold.

Data set	Model	$\Delta\bar{Y}$	$\sigma(\Delta Y)$	$\Delta\bar{m}_\phi$	$\sigma(\Delta m)_\phi$	$\Delta\bar{m}_\theta$	$\sigma(\Delta m)_\theta$	$\Delta\bar{m}_\psi$	$\sigma(\Delta m)_\psi$
D_1	1†	1.023e−1	–	**1.321**	–	6.327	–	**9.200e−2**	-
	2‡	3.867e−3	2.424e−4	22.669	4.603	25.309	4.626	1.402	0.466
	3	3.775e−3	3.525e−4	12.507	1.666	**1.706**	1.498	0.118	9.611e−2
	4	**3.493e−3**	3.296e−4	16.550	2.517	23.543	4.238	0.463	0.204
	5‡	2.625e−3	1.160e−4	26.409	4.746	**28.281**	5.491	0.499	0.118
	6	**2.480e−3**	1.030e−4	**24.337**	3.555	28.604	4.618	**0.273**	0.0677
	7‡	4.874e−3	6.764e−4	33.717	4.994	22.997	4.762	0.347	0.142
	8	**3.793e−3**	3.444e−4	**6.550**	1.546	**6.472**	1.478	**0.136**	4.227e−2
	9‡	5.981e−3	1.190e−3	40.489	7.631	23.196	6.782	**0.269**	0.141
	10	**4.405e−3**	6.185e−4	**12.053**	1.176	**1.171**	1.383	0.316	3.021e−2
D_2	11‡	2.747e−4	4.570e−5	**4.582**	1.829	**1.615**	0.796	57.040	11.849
	12	**2.580e−4**	4.290e−5	5.662	2.031	1.724	0.631	**23.399**	5.521
D_3	13‡	4.771e−3	3.661e−4	9.526	4.686	18.904	5.065	6.954	0.997
	14	**4.525e−3**	3.087e−4	**5.754**	3.315	**16.587**	3.202	**1.729**	0.716

indicator of how well the network learns the prior knowledge, which in our case is the conservation law of momentum.

Discussion 1: Linearized mathematical model as baseline. We first examined the performance given by the linearized mathematical model in Eq. (3) ("Model 1" in Table 3) compared to a vanilla DNN ("Model 2") and PINNs ("Model 3 and 4") with D_1 data set. It was found that both vanilla DNN and PINN, no matter with which settings, outperforms significantly the linearized model in terms of prediction error by at least one order of magnitude. This is expected since drag is included during the data collection which becomes non-negligible for agile maneuvers, thereby degrading the performance of linearized model. Interestingly, it is noted that linearized model has relatively small values on Δm, meaning that it is "in general" more physically consistent compared to those learning-based models. Despite sounding somewhat contradictory, it is understandable because CCE extracts linearity from the data to reveal the global trend, which is more robust to aerodynamic-dominant data samples than MSE. As a result, the linearized model has a larger "bias" on prediction, but scores a lower "bias" in terms of the slope of CCE.

Discussion 2: Effectiveness of PINN and cyclical annealing scheduler. Next, we draw the conclusion that PINN ("Model 3 and 4") has superior performance than the vanilla DNN ("Model 2") in terms of prediction error (for better visualization, see Fig. 7). Moreover, it can be seen that the use of cyclical annealing scheduler ("Model 4" with settings in Fig. 3) further boosts the performance compared to the one with constant value of regularizer ("Model 3"). We also plot 3σ CCE, as post-hoc model interpretability technique, of the predictions given by vanilla DNN and PINN for one single seed, which have competitive prediction performance on test error: 3.322×10^{-3} for the former while 3.061×10^{-3} for the latter. From Fig. 8, it can be seen that the prediction CCE of vanilla DNN has larger deviation of slope (or rotation angle) compared to the target CCE than PINN does, which implies less capability of learning prior knowledge and physical consistency of the learned model.

Discussion 3: Benefitting from BN. As an effective practice to enhance network optimization by eliminating internal covariate shifts within each layer, we also studied if PINN can take advantage of BN (note that "Model 2 to 4" do not use BN). From Table 3 or Fig. 7, it can be seen that we achieved the best model ("Model 6") for D_1 data set using PINN with both cyclical annealing scheduler and BN. Note that vanilla DNN also reaches competitive performance with BN ("Model 5") and we believe both two models in this situation are near performance saturation thanks to the large data set and the powerful learning potentials given by DNN and practical tricks. However, as we will see next, PINN establishes noticeable superiority over vanilla DNN when facing relatively smaller data set.

Discussion 4: Performance on reduced data set. To examine the advantage of PINN in terms of generalization capability on unseen data endowed by informed physics, we re-trained the vanilla DNN and PINN on the reduced data set, i.e., 20% (∼ 3k) of D_1 instead of 60% (∼ 9k). In this circumstance, PINN ("Model 8 and 10") has shown pronounced

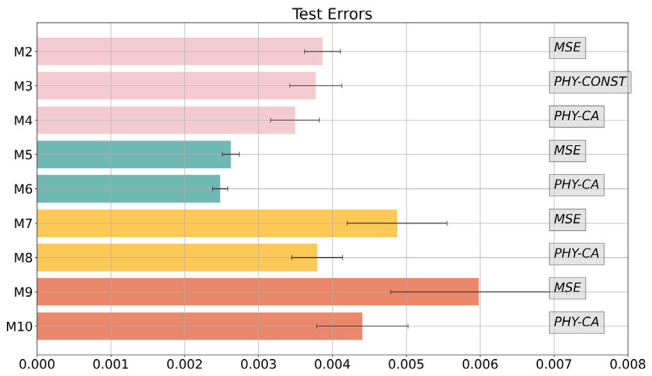


Fig. 7. Comparisons of test error by different model on D_1 data set. Vertical axis represents different models M2 to M10 with abbreviations of the employed loss function indicated in gray boxes on the right: "MSE" denotes the vanilla DNN, "PHY-CONST" denotes the PINN with constant hyperparameter ($\lambda_{\text{LM}} = \lambda_{\max}$), and "PHY-CA" denotes the PINN with cyclical annealing scheduler. Results in red bars are yielded from models (M2, M3, M4) trained on 60% of data (∼ 9k) without BN. Results of similar models but trained with BN ($M5, M6$) are shown in green. Yellow (M7, M8) and orange (M9, M10) bars show the results of models with BN but trained only on 20% of data (∼ 3k) and evaluated on 40% (∼ 6k) and 80% (∼ 12k) of data, respectively. All the results are averaged over multiple seeds with error bars representing the standard deviation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

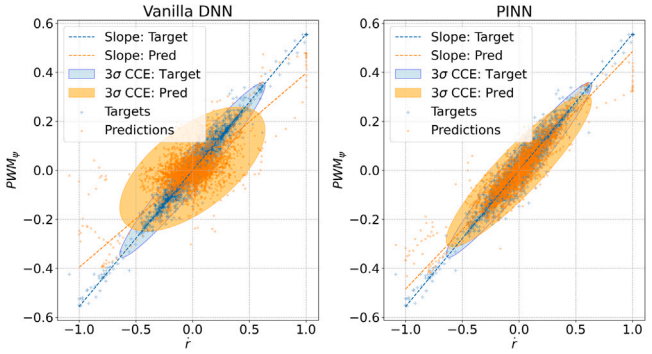


Fig. 8. Comparisons of CCE between vanilla DNN and PINN. Targets and predictions are indicated by dot and plus sign, respectively. Shaded ellipses illustrate 3σ CCEs with dashed line highlighting the slope. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

performance over vanilla DNN ("Model 7 and 9") in terms of both prediction error and physical consistency, though performance degradation is observed for both networks compared to those trained on the original (full) data set. This result highlights the stronger generalization capability of PINN in the small data regime. Hence, our proposed method constitutes an effective means to train DNN-based controllers with small data sets, while maintaining an acceptable generalization capability — an issue that is crucial in many robotics and control applications [38]. Meanwhile, it may also lay the foundation for online modeling using real flight data as our future research direction.

Discussion 5: Evaluation on data sets with ground effect and periodic wind. Lastly, we evaluated the vanilla DNN and PINN on D_2 (with additional ground effect) and D_3 (with additional periodic wind) data set. Same network structure was used except that we took altitude as an extra input feature for both networks trained with D_2 since it has been empirically proved to be a critical variable for ground effect [33,34]. From Fig. 9 and Table 3, it is shown that PINN has lower prediction error on both data sets and reveals higher physical consistency in terms of CCEs. Therefore, despite the fact that we guide the network learning with the conservation law of momentum instead of prior knowledge on aerodynamics, the resulting PINN exhibits better

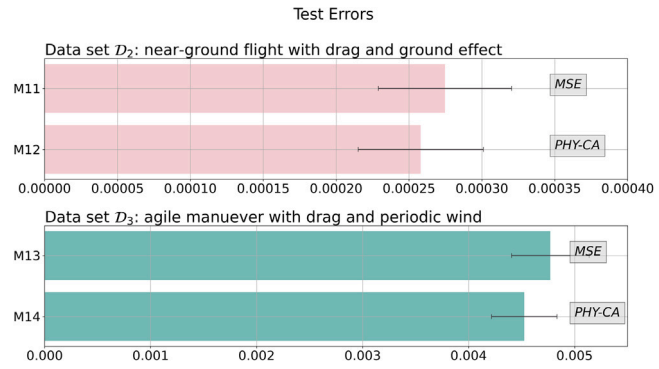


Fig. 9. Comparisons of test error by different model on D_2 (upper) and D_3 (lower) data set. All the results are averaged over multiple seeds with error bars representing the standard deviation.

generalization capability than vanilla DNN when facing different or even combined aerodynamic effects.

5.4. Computational complexity

Compared to the vanilla DNN, the additional computational complexity of our proposed PINN mainly comes from the inclusion of the local monotonicity loss (i.e., Algorithm 1), which is composed of $\tanh(\cdot)$ and other basic arithmetic operators. Thanks to automatic differentiation by PyTorch (e.g., "torch.autograd"), backpropagation using the chain rule of the known gradients can be easily performed due to the fact that we implemented Algorithm 1 all in form of tensors.

According to our training trials over 20 runs with random seed, given the training dataset of size ∼ 12k (∼ 9k for training, ∼ 3k for validation) and running for 300 epochs, it took ∼ 234 sec (∼ 3.9 min) and ∼ 373 sec (∼ 6.2 min) for training a vanilla DNN ("Model 2" in Table 3) and the proposed PINN ("Model 4" in Table 3), respectively, using CUDA on a laptop with AMD Ryzen 7 5800 h and NVIDIA GeForce RTX 3060. This increment of the training time is partially compensated by the fact that PINNs generally achieve better performance to vanilla DNNs in small data regime [15]. Undoubtedly, an online implementation may be unpractical for both methodologies. However, their usage is generally acceptable for non-time-critical situations, where offline training is allowed.

6. Conclusions

Learning-based approaches such as DNN have aroused unprecedented attention in modeling and data-driven control in recent years. Nonetheless, only few efforts have been paid to address the interpretability issue of these black-box methods, thereby leading to major concerns in safety and trustworthiness when applied to robotic applications in the real world. In this work, we step toward this goal by proposing a PINN for learning quadrotor dynamics, which in essence is a deep neural network informed by prior domain knowledge. We showed in detail how to seamlessly embed the conservation law of momentum into the training loss function with cyclical annealing scheduler following an unstructured learning strategy. Besides the physics-informed model, we also introduced CCE as a post-hoc model interpretability visualization to evaluate and understand model's behavior. All of these are devoted to enhancing the trustworthiness of learning-based approach.

To facilitate data collection in our studies, we also set up a visual and physical simulator based on AirSim with a custom implementation of the quadrotor ground effect. User-level APIs in C++ and Python are provided, which are all made publicly available for further research in UAV community. Through our extensive simulation campaign over

multiple seeds, our proposed PINN consistently demonstrates better generalization capability compared to linearized mathematical model and vanilla DNN evaluated on both complete and reduced data set including various aerodynamics. We also carried out ablation studies and analyzed the additional computational complexity of our proposed approach compared to vanilla DNN.

As for future work, we aim to continue exploring the potential of PINN for online learning using real flight data. Besides, we plan to integrate the proposed PINN as control-oriented model into the controller design to further evaluate its contribution to the closed-loop performance. Last but not least, we would like to investigate new paradigms of PINN that combine structured and unstructured learning so as to leverage their own advantages for UAV modeling and control.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Weibin Gu reports financial support was provided by Amazon Science. Alessandro Rizzo reports financial support was provided by PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR).

Data availability

Data will be made available on request.

Acknowledgments

This work has been carried out within the MOST – Sustainable Mobility National Research Center and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1033 17/06/2022, CN00000023) and within the FAIR - Future Artificial Intelligence Research and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.3 – D.D. 1555 11/10/2022, PE00000013). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them. Weibin Gu is partially supported by the 2021 Amazon Research Award titled "Physics-Informed Machine Learning for Trustworthy Control of Autonomous Robots."

References

[1] K. Takaya, H. Ohta, K. Shibayama, V. Kroumov, Tracking control of unmanned aerial vehicle for power line inspection, Motion Planning [Working Title] (2021).
[2] W. Gu, D. Hu, L. Cheng, Y. Cao, A. Rizzo, K.P. Valavanis, Autonomous wind turbine inspection using a quadrotor, in: 2020 International Conference on Unmanned Aircraft Systems, ICUAS, 2020, pp. 709–715.
[3] N. Dalwadi, D. Deb, S.M. Muyeen, Adaptive backstepping controller design of quadrotor biplane for payload delivery, IET Intell. Transp. Syst. (2022).
[4] H. Shakhatreh, A.H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E.K. Almaita, I.M. Khalil, N.S. Othman, A. Khreishah, M. Guizani, Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges, IEEE Access 7 (2019) 48572–48634.
[5] W. Gu, K.P. Valavanis, M.J. Rutherford, A. Rizzo, UAV model-based flight control with artificial neural networks: A survey, J. Intell. Robotic Syst. 100 (2020) 1469–1491.
[6] P.A. Ioannou, J. Sun, Robust adaptive control, 2012.
[7] G. Hoffmann, H. Huang, S.L. Waslander, C.J. Tomlin, Precision flight control for a multi-vehicle quadrotor helicopter testbed, Control Eng. Pract. 19 (2011) 1023–1036.
[8] T.-W. Ou, Y. Liu, Adaptive backstepping tracking control for quadrotor aerial robots subject to uncertain dynamics, in: 2019 American Control Conference, ACC, 2019, pp. 1–6.
[9] T. Chingozha, O.T. Nyandoro, Adaptive sliding backstepping control of quadrotor UAV attitude, IFAC Proc. Vol. 47 (2014) 11043–11048.
[10] A. Punjani, P. Abbeel, Deep learning helicopter dynamics models, in: 2015 IEEE International Conference on Robotics and Automation, ICRA, 2015, pp. 3223–3230.
[11] S. Bansal, A.K. Akametalu, F.J. Jiang, F. Laine, C.J. Tomlin, Learning quadrotor dynamics using neural network for flight control, in: 2016 IEEE 55th Conference on Decision and Control, CDC, 2016, pp. 4653–4660.
[12] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, S.-J. Chung, Neural lander: Stable drone landing control using learned dynamics, in: 2019 International Conference on Robotics and Automation, ICRA, 2019, pp. 9784–9790.
[13] G. Shi, W. Honig, X. Shi, Y. Yue, S.-J. Chung, Neural-Swarm2: Planning and control of heterogeneous multirotor swarms using learned interactions, 2020, ArXiv. arXiv:2012.05457.
[14] A.B. Arrieta, N.D. Rodríguez, J.D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, F. Herrera, Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, 2019, ArXiv. arXiv:1910.10045.
[15] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, 2021.
[16] J. Callanan, R. Iqbal, R. Adlakha, A. Behjat, S. Chowdhury, M. Nouh, Large-aperture experimental characterization of the acoustic field generated by a hovering unmanned aerial vehicle, J. Acoust. Soc. Am. 150 3 (2021) 2046.
[17] R. Iqbal, A. Behjat, R. Adlakha, J. Callanan, M. Nouh, S. Chowdhury, Efficient training of transfer mapping in physics-infused machine learning models of UAV acoustic field, 2022, ArXiv arXiv:2201.06090.
[18] R. Iqbal, A. Behjat, R. Adlakha, J. Callanan, M. Nouh, S. Chowdhury, Auto-differentiable transfer mapping architecture for physics-infused learning of acoustic field, IEEE Trans. Artif. Intell. (2023).
[19] M. Lutter, C. Ritter, J. Peters, Deep Lagrangian networks: Using physics as model prior for deep learning, 2019, ArXiv. arXiv:1907.04490.
[20] J.K. Gupta, K. Menda, Z. Manchester, M.J. Kochenderfer, A general framework for structured learning of mechanical systems, 2019, ArXiv arXiv:1902.08705.
[21] M. Lutter, J. Peters, Combining physics and deep learning to learn continuous-time dynamics models, 2021, ArXiv arXiv:2110.01894.
[22] A. Karpatne, W. Watkins, J.S. Read, V. Kumar, Physics-guided neural networks (PGNN): An application in lake temperature modeling, 2017, ArXiv arXiv:1710.11431.
[23] S. Shaier, M. Raissi, P. Seshaiyer, Data-driven approaches for predicting spread of infectious diseases through DINNs: Disease informed neural networks, 2021.
[24] S. Shah, D. Dey, C. Lovett, A. Kapoor, AirSim: High-fidelity visual and physical simulation for autonomous vehicles, in: FSR, 2017.
[25] C. Schelp, An alternative way to plot the covariance ellipse, 2018, https://carstenschelp.github.io/2018/09/14/Plot_Confidence_Ellipse_001.html.
[26] J.E. Sierra, M. Santos, Modelling engineering systems using analytical and neural techniques: Hybridization, Neurocomputing 271 (2018) 70–83.
[27] L. Bauersfeld, E. Kaufmann, P. Foehn, S. Sun, D. Scaramuzza, NeuroBEM: Hybrid aerodynamic quadrotor model, 2021, ArXiv arXiv:2106.08015.
[28] M. O'Connell, G. Shi, X. Shi, K. Azizzadenesheli, A. Anandkumar, Y. Yue, S.-J. Chung, Neural-fly enables rapid learning for agile flight in strong winds, Science Robotics 7 (2022).
[29] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[30] S. Greydanus, M. Dzamba, J. Yosinski, Hamiltonian neural networks, 2019, ArXiv arXiv:1906.01563.
[31] Y.D. Zhong, B. Dey, A. Chakraborty, Symplectic ODE-net: Learning Hamiltonian dynamics with control, 2020, ArXiv arXiv:1909.12077.
[32] A. Karpatne, G. Atluri, J.H. Faghmous, M.S. Steinbach, A. Banerjee, A.R. Ganguly, S. Shekhar, N.F. Samatova, V. Kumar, Theory-guided data science: A new paradigm for scientific discovery from data, IEEE Trans. Knowl. Data Eng. 29 (2017) 2318–2331.
[33] S.A. Conyers, M.J. Rutherford, K.P. Valavanis, An empirical evaluation of ground effect for small-scale rotorcraft, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, 2018, pp. 1244–1250.
[34] X. Kan, J.R. Thomas, H. Teng, H.G. Tanner, V.R. Kumar, K. Karydis, Analysis of ground effect for small-scale UAVs in forward flight, IEEE Robot. Autom. Lett. 4 (2019) 3860–3867.
[35] R.W. Beard, Quadrotor dynamics and control, 2008.
[36] H. Fu, C. Li, X. Liu, J. Gao, A. Celikyilmaz, L. Carin, Cyclical annealing schedule: A simple approach to mitigating kl vanishing, in: NAACL, 2019.
[37] I.C. Cheeseman, P. D, W.E. Bennett, P. D, W.E. Bennett, The effect of Ground on a Helicopter Rotor in Forward Flight, Tech. Rep., 1955.
[38] G. Joshi, J. Virdi, G.V. Chowdhary, Asynchronous deep model reference adaptive control, 2020, ArXiv arXiv:2011.02920.

Weibin Gu is currently a Ph.D. candidate in the Department of Electronics and Telecommunications at Politecnico di Torino. He received his M.Sc. Degree in Mechatronic Engineering from Politecnico di Torino in 2017 and his B.Sc. Degree in Mechanical Engineering from Tongji University and Politecnico di Torino (dual-degree program) in 2015. From September 2019 to October 2020, he worked as flight control engineer at Shanghai FOIA Co., Ltd., China, responsible for UAV-based autonomous project for industrial applications. He has also been actively collaborating with the University of Denver and Politecnico di Torino since July 2017. His research focuses on the intersection of machine learning and control theory with the applications to real-world complex systems such as robotics.

Stefano Primatesta is currently an Assistant Professor in the Department of Mechanical and Aerospace Engineering. He received his Ph.D. in Computer and Control Engineering from Politecnico di Torino in 2019, his M.Sc. in Mechatronic Engineering, and his B.S in Electronic Engineering from Politecnico di Torino in 2011 and 2014, respectively.

His field of research is the use of Remotely Piloted Aircraft Systems in urban environments including virtual modeling and multi-dimensional risk analysis. His research interests include also autonomous navigation and service robotics, with applications on unmanned aerial vehicles and unmanned ground vehicles.

Alessandro Rizzo received the Laurea degree (summa cum laude) in computer engineering and the Ph.D. degree in automation and electronics engineering from the University of Catania, Italy, in 1996 and 2000, respectively. In 1998, he worked as a EURATOM Research Fellow with JET Joint Undertaking, Abingdon, U.K., researching on sensor validation and fault diagnosis for nuclear fusion experiments. In 2000 and 2001, he has worked as a Research Consultant at ST Microelectronics, Catania Site, Italy, and as an Industry Professor of robotics with the University of Messina, Italy.

From 2002 to 2015, he was a tenured Assistant Professor with the Politecnico di Bari, Italy. Since 2012, he has been a Visiting Professor with the New York University Tandon School of Engineering, Brooklyn, NY, USA.

In November 2015, he joined Politecnico di Torino, where he is an Associate Professor in the Department of Electronics and Telecommunications and established the Complex Systems Laboratory. Dr. Rizzo is engaged in conducting and supervising research on complex networks and systems, modeling and control of nonlinear systems, and cooperative robotics. He is the author of one book, two international patents, and about 200 papers on international journals and conference proceedings. He was a recipient of the Award for the Best Application Paper at the IFAC world triennial conference in 2002 and of the Award for the Most Read Papers in Mathematics and Computers in Simulation (Elsevier) in 2009. He has also been a Distinguished Lecturer of the IEEE Nuclear and Plasma Science Society and one of the recipients of the 2019 and 2021 Amazon Research Awards in robotics.