Communications on Applied Mathematics and Computation
https://doi.org/10.1007/s42967-025-00499-x

ORIGINAL PAPER

Physics-Informed Neural Networks for PDE-Constrained Optimization and Control

Jostein Barry-Straume[1] · Arash Sarshar[2] · Andrey A. Popov[3] · Adrian Sandu[1]

Received: 3 October 2024 / Revised: 7 March 2025 / Accepted: 10 April 2025

Abstract
The goal of optimal control is to determine a sequence of inputs for maximizing or minimizing a given performance criterion subject to the dynamics and constraints of the system under observation. This work introduces Control Physics-Informed Neural Networks (PINNs), which simultaneously learn both the system states and the optimal control signal in a single-stage framework that leverages the system's underlying physical laws. While prior approaches often follow a two-stage process-modeling, the system first and then devising its control—the presented novel framework embeds the necessary optimality conditions directly into the network architecture and loss function. We demonstrate the effectiveness of the novel methodology by solving various open-loop optimal control problems governed by analytical, one-dimensional, and two-dimensional partial differential equations (PDEs).

Keywords Optimal control · Scientific machine learning (SciML) · Physics-informed neural networks (PINNs)

Mathematics Subject Classification 65K05 · 35Q93 · 90C46

✉ Jostein Barry-Straume
jostein@vt.edu

Arash Sarshar
arash.sarshar@csulb.edu

Andrey A. Popov
apopov@hawaii.edu

Adrian Sandu
sandu@cs.vt.edu

[1] Department of Computer Science, Virginia Polytechnic and State University, Blacksburg, VA 24060, USA

[2] Department of Computer Engineering and Computer Science, California State University, Long Beach, CA 90840, USA

[3] Department of Information and Computer Sciences, The University of Hawai'i at Mānoa, Honolulu, HI 96822, USA

Published online: 18 August 2025

1 Introduction

Scientific machine learning (SciML) has gained momentum as an alternative and a complement to traditional numerical methods. The main driving force behind SciML is the success of neural networks in regression, classification, and generative tasks [2]. In their systematic review, Willard et al. [46] gave a structured overview of ML-based approaches to physics-based modeling and listed the current areas where SciML is being used. When it comes to approximating solutions to partial differential equations (PDEs), neural networks are a compelling and practical option because of their mesh-free outputs and the availability of derivatives via automatic differentiation [11, 22, 37].

Physics-informed neural networks (PINNs) [34] solve semi-supervised learning tasks while respecting physical laws. This is accomplished by incorporating the underlying mathematical equations into the loss function. Raissi et al. utilized PINNs for solving physical equations, and for data-driven discovery of PDEs [34]. The general procedure for solving a differential equation with a PINN involves tuning the parameters of a network that minimize a loss function. The loss function includes the difference between the output and the data, as well as the boundary and initial conditions' residuals, PDE equations, and any other physical constraints that are needed [22]. The recent survey by Cuomo et al. [9] offers a thorough examination of PINNs and addresses various customizations, including distinct activation functions, gradient optimization methods, neural network architectures, and loss functions. Since their inception PINNs have been applied to solve a wide range of problems including PDEs [19, 34, 47, 49], inverse problems [6, 15, 22, 32, 33, 35], solution of fractional differential equations [31], and stochastic differential equations [27, 48, 50, 51].

Optimal control problems are discrete or continuous decision-making problems in which a control signal is applied to the given system over time and space to satisfy certain optimality constraints [3]. The sequential decisions can be thought of as a control policy that, given an initial state of the system, efficiently enacts decisions such that an objective is optimized (e.g., minimizing transition time to a desired state, or maximizing profit) [3]. Knowledge of both the system state and the effects of the control is necessary to develop an optimal control policy via learning through experience.

Some recent work has begun exploring the application of PINNs to solve optimal control problems [1, 7, 18, 24, 43]. However, these existing approaches either feed control data to a PINN, thereby training on precomputed control signals, or using an external control mechanism in conjunction with the PINN model of the physical system; in other words, an outer loop optimization with an inner loop PINN used as a surrogate model.

This paper introduces a new framework called Control PINNs, which is designed to solve open-loop optimal control problems. Control PINNs can simultaneously determine the system state, the adjoint system state, and the optimal control signal without relying on preexisting controller data or an external controller. In essence, Control PINNs can find optimal control solutions for challenging computational scientific problems without separating the learning tasks of the system state and the control signal into different frameworks.

The remainder of the paper is organized as follows: Sect. 2 delves into the advantages and disadvantages of one-stage versus two-stage frameworks in learning control variables for PDEs. Section 3 offers a survey of current approaches in optimal control with PINNs. Section 4 covers the novel methodology of this paper. In Sect. 4.1, we discuss software implementation aspects of our methodology. Section 5 validates the methodology via the implementation of an analytical problem. Section 6 presents experimental results for controlling a one-dimensional unsteady heat equation. Section 7 offers a more challenging optimal

control problem for a two-dimensional reaction-diffusion problem. Section 8 summarizes the results and details the groundwork for future directions.

2 Open-Loop Control Problems

There are two main approaches to solving open-loop optimal control problems [4]. The first is nested analysis and design (NAND), a black-box, multi-stage approach. The steps are: (i) begin with an initial guess for the optimal control $\mathbf{u}$ and optimal solution $\mathbf{y}$, (ii) solve the forward equations and assess the loss function, (iii) solve the adjoint equations to find the gradient with respect to controls, (iv) update the controls using an optimization method, and (v) repeat the process with the new controls.

This approach offers the benefit of comparatively straightforward implementation. The disadvantages include the need to resolve the model equations at each iteration of the optimization algorithm to determine the state and adjoint. It is also possible that the optimizer can generate non-physical intermediate values of $\mathbf{y}$ and $\mathbf{u}$ through the indirect relationship between the decision variables and the state constraints. This approach can be quite inefficient for systems in which the model equations are expensive to solve and the states may be highly constrained. When the forward model is replaced by a PINN, it must be retrained at every iteration of the optimization, as each iteration corresponds to a different operation point (different control value $\mathbf{u}$ and solution $\mathbf{y}$).

The second approach is simultaneous analysis and design (SAND), an all-at-once, one-stage approach, where both controls and states are part of the decision variable space. The optimal values of the control $\mathbf{u}$, solution $\mathbf{y}$, and adjoint $\boldsymbol{\lambda}$ are directly computed, avoiding possible non-physical intermediate values. This approach results in large systems of nonlinear equations that couple the forward, adjoint, and controls at every intermediate time step across all spatial locations. In contrast, the numerical NAND approach is more economical as it does not couple variables at different times.

The comparative benefits and drawbacks of the two methodologies shift when employing ML surrogates in place of numerical solutions. PINNs employed in a NAND methodology require training during each optimization iteration. On the other hand, we have designed Control PINNs to follow the SAND approach and solve the forward PDE, the adjoint, and the constraints in continuous time and space simultaneously. The coupled system of equations is satisfied efficiently by minimizing the expected value of the PINN loss function using a stochastic optimizer such as `Adam`.

3 Survey of Current Approaches in Optimal Control with PINNs

Chen et al. trained an input convex recurrent neural network and subsequently solved a convex model predictive control (MPC) problem on the learned model [7]. The main strength of this approach is the guarantee of an optimal solution, due to the convex nature of the model. The main limitation is similar to [18], in that they employ a two-stage framework of system identification and controller design. Success is evaluated on four different experiments conducted in the paper where the input convex recurrent neural network results are compared to that of a standard multilayer perceptron (MLP). The control action is also compared to that of the baselines of conventional optimizations.

Antonelo et al. introduced a new framework called Physics-Informed Neural Nets for Control (PINC) [1]. PINC uses data from the control action and initial state to solve an optimal control problem. One strength of this approach is the ability to make predictions beyond the training time horizon for an indefinite period without a significant reduction in prediction capability. A limitation of this approach is offline learning the control separately from the solution operator. In other words, PINC is essentially a PINN that is amenable to being trained on the actions of an external controller, instead of learning the optimal control unsupervised. Success is evaluated through the Mean Squared Error (MSE) validation error of the solution to the Van der Pol Oscillator problem.

Nellikkath and Chatzivasileiadis [28] showed that combining Karush-Kuhn-Tucker (KKT) conditions with a PINN achieves higher accuracy while utilizing substantially fewer data points. The authors demonstrated success by training a PINN to solve for DC Optimal Power Flow. In addition, they showed that such PINNs commit fewer worst-case violations than conventional neural networks.

Wang et al. leveraged physics-informed DeepONets "as a fast and differentiable surrogate for tackling high-dimensional PDE-constrained optimization problems via gradient-based optimization in near real-time" [43]. The foundational idea behind their approach is to optimize a network that associates an outcome with a set of controllable variables. The strength of the approach comes from leveraging DeepONets in a physics-informed fashion. This allows smaller training datasets, as their framework makes for an effective emulator. The limitation of their approach involves learning the control separately from the system state. The paper proposes sequentially training a neural network to learn the solution operator of a given PDE system, and then passing that information to another neural network to learn to associate the input system state with a certain control action. This approach, like others mentioned previously, is a two-stage framework. Moreover, the challenge of incorporating adjoint equations into their framework is circumvented, where indeed there is empirical evidence in favor of using adjoint information [8]. Here, one measure of performance considered is the training time of PINNs compared to that of traditional numerical solvers. This benchmarking is conducted for both optimal control of heat transfer, and drag minimization of obstacles in the Stokes flow. Moreover, a numerical solver is utilized to validate and test the inferred control solution versus the found solution.

Hwang et al. proposed a two-stage framework for solving PDE-constrained control problems using operator learning [18]. They first trained an autoencoder model, and then inferred the optimal control by fixing the learnable parameter and minimizing their objective function. One strength of their approach is the ability to apply their framework to both data-driven and data-free cases. The main downside to their approach is the two-stage nature of the framework, as the control is found only after a surrogate model has been trained. Success is measured by tracking the relative error against numerical simulation in the case of data-driven experiments, and the relative error against an analytical solution in the case of data-free experiments. Additionally, the visual inspection of the trained solution operators is conducted.

In [41], Shah et al. presented a fractal-fractional epidemiological framework to model the spread of a generic viral infectious disease in the presence of a vaccination program. The authors embedded fractal functions and their derivatives in the definition and computation of the time of evolution for each "compartment" of their model. In doing so, their model is enabled to better mimic the nuanced features of the system dynamics of viral transmission.

In [39], Shah et al. put forth a coupled piecewise modeling framework for describing and analyzing the kinetics of drug treatments when the administration is naturally piecewise in time. Instead of a single, continuous set of differential equations, the authors split the model into different "phases" or time intervals, enabling them to capture piecewise changes in

dosage or therapy strategies more accurately. The presented novel methodology is a system of fractional differential equations, where each equation or group of equations describes different treatment stages of drug administration.

In [40], Shah et al. displayed the usefulness of fractal-fractional analysis in epidemiological modeling. The authors showed that combining fractional calculus with fractal concepts can yield more robust and realistic predictions of disease dynamics, especially in situations where the disease exhibits long-term memory or complex transmission patterns. Specifically, the model's derivative with respect to time is replaced with a fractal-fractional operator that depends on two parameters: (i) a fractional order α which captures disease memory dynamics, and (ii) a fractal dimension β that captures fractal, self-similar propagated behavior or scale-invariant effects.

Mowlavi and Nabi conducted an evaluation of the comparative performance between traditional PINNs and classic direct-adjoint-looping (DAL) to solve optimal control problems [24]. Their optimal control problem is separated into two subproblems. At each state of the system, the PDE is solved with one neural network. That information is then used by another neural network to solve for the optimal control at that given state of the PDE. Afterward, the adjoint PDE is solved in backward time. The strength of Mowlavi and Nabi's approach is providing an evaluative comparison between PINNs and DAL frameworks for solving PDE-constrained optimal control problems. Their comparison provides a frame of reference for PINNs. Success is measured via validation and evaluation steps. Validation is done by monitoring residual, boundary, and initial loss components with a known solution. Evaluation is done by comparing the control cost objective with a solution found by a high-fidelity numerical solver. The manual derivation is used in their DAL approach, which is unnecessary because DAL can use automatic differentiation (AD). Consequently, to a certain extent, the optimal control problem is being solved manually. Furthermore, the control of the system is being dampened over time. It should be noted that the adjoint PDE is not being solved in their cost function, and thus the respective adjoint formulas are not present in the said cost function. This brings us to the main contributions of this paper.

The approaches put forth by [1, 7, 18, 24, 43] leverage PINNs as surrogate models for control optimization. In contrast, we propose a one-stage framework that integrates PDE-constrained optimization directly into the training process, simultaneously learning the system solution and optimal control.

4 Methodology

We begin by considering an abstract formulation of a PDE-constrained control problem [14]:

$$\begin{cases} \mathbf{u}^* = \arg\min_{\mathbf{u}} \Psi(\mathbf{u}, \mathbf{y}), \\ \Psi(\mathbf{u}, \mathbf{y}) = \displaystyle\int_{t_0}^{t_f} \int_{\Omega} g(\mathbf{y}, \mathbf{u}) \, \mathrm{d}x\mathrm{d}t + \int_{\Omega} w(\mathbf{y}|_{t_f}) \mathrm{d}x + \int_{t_0}^{t_f} \int_{\partial\Omega} h(\mathbf{y}, \mathbf{u}) \, \mathrm{d}x\mathrm{d}t, \\ \text{subject to the PDE:} \quad \begin{cases} \frac{\partial \mathbf{y}}{\partial t} = \mathbf{f}(\mathbf{y}, \mathbf{u}), & \forall t \in [t_0, t_f], \ \forall x \in \Omega, \\ \mathbf{y}|_{t_0} = \mathbf{y}_0, & \forall x \in \Omega, \\ \mathcal{B}\mathbf{y} = \mathbf{b}(t, x), & \forall t \in [t_0, t_f], \ \forall x \in \partial\Omega, \end{cases} \end{cases} \tag{1}$$

where $\mathbf{y} = \mathbf{y}(t, x)$ is a time- and space-dependent system state and $\mathbf{u} = \mathbf{u}(t, x)$ is a time- and space-varying control signal applied to the system. We are interested in minimizing a given objective functional Ψ that may depend on the solution, control, or both while also satisfying the physical laws prescribed by the PDE. $\mathbf{f}$ is the right-hand-side of the PDE and describes the

time-derivative of $\mathbf{y}$. $\mathcal{B}$ and $\mathbf{b}$ define the boundary conditions, and $\mathbf{y}_0$ is the initial condition of the system state.

The objective function Ψ is very flexible and can be defined over the physical domain, time, boundaries, and desired final state of the system: g is a cost over the spatio-temporal domain, w is a cost at the final time, and h is a cost over the boundary. For simplicity, this work assumes that h is zero everywhere. The functions $\mathbf{y}$ and $\mathbf{u}$ live in appropriate function spaces with sufficient smoothness that ensure the existence of the optimal solution [16, Sect. 1.5]. The PDE is defined over domain Ω with the boundary $\partial\Omega$.

Next we define the Lagrangian of Eq. (1) as

$$\mathfrak{L} = \int_{t_0}^{t_f} \int_{\Omega} \left[g\left(\mathbf{y}, \mathbf{u}\right) - \boldsymbol{\lambda}^{\mathrm{T}} \left(\frac{\partial \mathbf{y}}{\partial t} - f\left(\mathbf{y}, \mathbf{u}\right) \right) \right] \mathrm{d}x\mathrm{d}t + \int_{\Omega} w(\mathbf{y}t_f)\mathrm{d}x. \tag{2}$$

Notice the adjoint variable $\boldsymbol{\lambda}^{\mathrm{T}}(t, x)$ multiplying the residual of the PDE. First order optimality conditions [30] for the local minimum of Eq. (2) are

$$\begin{aligned}
&\frac{\partial \mathbf{y}}{\partial t} = \mathbf{f}\left(\mathbf{y}, \mathbf{u}\right), && \forall t \in [t_0, t_f], \forall x \in \Omega, && \text{(3a)}\\
&\mathbf{y}|_{t_0} = \mathbf{y}_0, && \forall x \in \Omega, &&\\
&\mathcal{B}\,\mathbf{y} = \mathbf{b}(t, x), && \forall t \in [t_0, t_f], \forall x \in \partial\Omega, &&\\
&\frac{\partial \boldsymbol{\lambda}}{\partial t} = -\boldsymbol{\lambda}^{\mathrm{T}} \frac{\partial \mathbf{f}}{\partial \mathbf{y}}\left(\mathbf{y}, \mathbf{u}\right) - \frac{\partial g}{\partial \mathbf{y}}\left(\mathbf{y}, \mathbf{u}\right), && \forall t \in [t_0, t_f], \forall x \in \Omega, && \text{(3b)}\\
&\boldsymbol{\lambda}|_{t_f} = w_{\mathbf{y}}(\mathbf{y}|_{t_f}), && \forall x \in \Omega, &&\\
&\mathcal{B}^{*}\boldsymbol{\lambda} = 0, && \forall t \in [t_0, t_f], \forall x \in \Omega, &&\\
&\boldsymbol{\lambda}^{\mathrm{T}} \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\left(\mathbf{y}, \mathbf{u}\right) + \frac{\partial g}{\partial \mathbf{u}}\left(\mathbf{y}, \mathbf{u}\right) = \mathbf{0}, && \forall t \in [t_0, t_f], \forall x \in \Omega. && \text{(3c)}
\end{aligned}$$

The conditions in Eq. (3) need to be satisfied point-wise over time and space. Equation (3a) represents the evolution of the system state, Eq. (3b) represents the evolution of the adjoint equation, and Eq. (3c) represents the push-back of the control onto the dynamical system.

We note that the adjoint variable $\boldsymbol{\lambda}$ couples all optimality conditions in Eq. (3), and hence, this is a one-stage approach as discussed in Sect. 2. Additionally, the adjoint $\boldsymbol{\lambda}$ provides the sensitivity of the objective function with respect to changes in the constraints [30], which can be used for diagnostics and system optimization.

As an illustrative example, in the case of autonomous cars, the equations of motion determine the system state, which includes the vehicle's position and velocity. The software that controls the steering wheel, acceleration, and braking is the system controller. Lastly, the vehicle's reaction to the direction and speed selections made by the software would be the adjoint, also known as the co-state variable.

4.1 Design of Control PINNs

We design a neural network that generates triples $(\mathbf{y}, \mathbf{u}, \boldsymbol{\lambda})$ from input data (t, x). We equip the network with a PINN loss function that captures the first-order optimality conditions according to Eq. (3). The process of solving the control problem Eq. (1) using a Control PINN model involves the following steps.

i. Construct a network with inputs t, x, and outputs $\mathbf{y}$, $\mathbf{u}$, and $\boldsymbol{\lambda}$ based on the architecture in Fig. 1.

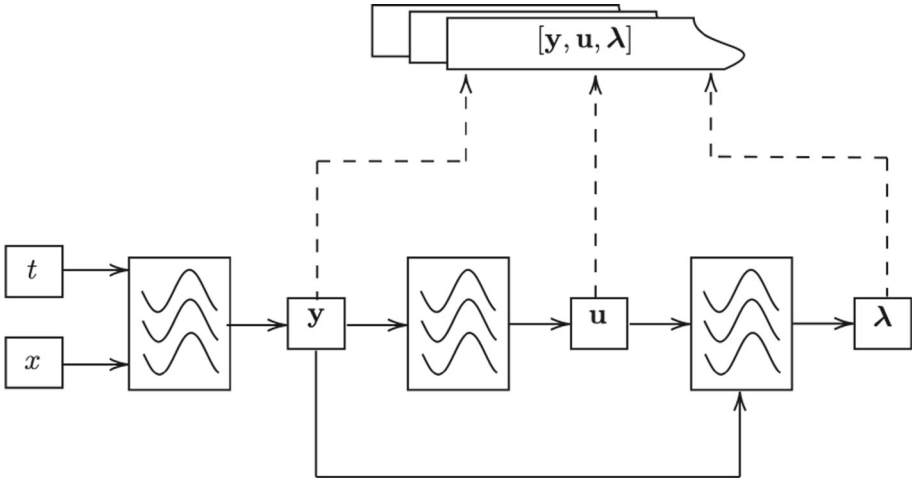


Fig. 1 Schematic of Control PINN architecture: time and space t, x are inputs to the model, and the system state $\mathbf{y}$, control $\mathbf{u}$, and the adjoint variable $\boldsymbol{\lambda}$ are the outputs of the model. The boxes with wave lines represent deep layers

ii. Compute the necessary derivatives of the outputs via backpropagation. These derivatives will be used to calculate the residuals in the loss function Eq. (4).

iii. With $\mathbf{y}_*^i$ denoting the training data for the input (t^i, x^i), and $\mathbf{y}^i, \mathbf{u}^i, \boldsymbol{\lambda}^i$ corresponding outputs of the network, $\boldsymbol{\theta}_k$ representing the parameters of the network at iteration k, and $\|\cdot\|$ as the L_2 norm, we seek to minimize the loss function:

$$\begin{aligned} L(\boldsymbol{\theta}_k) = &\sum_i \|\mathbf{y}_*^i - \mathbf{y}^i\|^2 \\ &+ \sum_i \left\| \frac{\partial \mathbf{y}^i}{\partial t} - f\left(\mathbf{y}^i, \mathbf{u}^i\right) \right\|^2 \\ &+ \sum_i \left\| \frac{\partial \boldsymbol{\lambda}^i}{\partial t} + (\boldsymbol{\lambda}^i)^{\mathrm{T}} \frac{\partial \mathbf{f}}{\partial \mathbf{y}^i}\left(\mathbf{y}^i, \mathbf{u}^i\right) + \frac{\partial g}{\partial \mathbf{y}^i}\left(\mathbf{y}^i, \mathbf{u}^i\right) \right\|^2 \\ &+ \sum_i \left\| \left(\boldsymbol{\lambda}^i\right)^{\mathrm{T}} \frac{\partial \mathbf{f}}{\partial \mathbf{u}^i}\left(\mathbf{y}^i, \mathbf{u}^i\right) + \frac{\partial g}{\partial \mathbf{u}^i}\left(\mathbf{y}^i, \mathbf{u}^i\right) \right\|^2. \end{aligned} \tag{4}$$

Note that the terms on this loss function are the residuals of Eq. (3). If boundary, initial, or final time data points are also available, their corresponding residuals from Eq. (3) can be added to the loss.

iv. Update the network parameters using the optimizer according to the loss function Eq. (4),

$$\boldsymbol{\theta}_{k+1} = \mathtt{ADAM}(\boldsymbol{\theta}_k, \nabla_{\boldsymbol{\theta}_k} L(\boldsymbol{\theta}_k)).$$

v. Repeat until convergence.

One limitation would be approximating chaotic systems over long time intervals. However, any given PINN faces this difficulty. Similarly, finding optimal controls for chaotic systems is difficult even for standard numerical methods. It should be noted that the adjoints of chaotic atmospheric dynamics are used successfully at the European Centre for Medium-Range Weather Forecasts (ECMWF) for data assimilation in numerical weather prediction.

Additionally, of note is the work of Wang and collaborators in alternative definitions of adjoints for chaotic systems [5]. The scope of Control PINN is mainly concerned with controls for non-chaotic systems, as is typical in engineering systems.

The main technical challenge is learning the state of a dynamical system while at the same time finding its optimal control. We create a path in the computational graph for the backpropagation of derivatives by placing the deep layers that generate $\mathbf{u}$ subsequent to the layers that generate $\mathbf{y}$ and, similarly, for $\boldsymbol{\lambda}$. This ensures that all necessary derivatives for Eq. (3) can be computed using automatic differentiation.

Analysis of convergence. With the overlap of fields of study comes an overlap of definitions. In numerical analysis, the term *convergence* refers to the behavior of numerical solutions as the discretization time step or grid size becomes small. In PINNs, there is no counterpart of a small parameter going to zero. Perhaps the convergence of a PINN could be observed via studying approximation errors as its dimension is increased. While the universal approximation property of NNs [17] has long ago been established, *convergence-type* results wherein the approximation error is bounded by a function of the number of neurons not yet generally available. Extending the concept of "convergence" to encompass PINNs is important to link the worlds of machine learning and numerical mathematics. Such an endeavor requires considerable additional investigation, and is beyond the scope of this paper.

Analysis of stability. With regard to differential equations, the stability analysis is the examination of how sensitive a solution is to small perturbations in initial conditions or parameters. Along those lines, a preliminary investigation into the stability of learned optimal solutions from the Control PINN architecture is presented in "Appendix B". In particular, Fig. B1 shows how different random weight initializations of the model lead to the same approximate learned solution of the optimal control signal of the one-dimensional heat equation problem in Sect. 6. Likewise, Fig. B2 shows similar results for the corresponding system dynamics. That being said, a fully fledged analysis of stability is beyond the scope of this paper. Future implementation would leverage the works presented in [25, 26, 38]. Specifically, in [38], the fixed point theory was implemented to establish conditions for which there exists at least one solution with respect to a class of hybrid differential equations. Moreover, in [26], a sensitivity analysis was conducted using the normalized forward sensitivity index method to quantify the impact of variations in each parameter on the bushfire transmission metric $\mathbb{R}_0$ in the context of cashew nut production. Additionally, in [25], a sensitivity study was carried out on the Maize Foliar Disease transmission dynamics in maize plants. The authors investigated the disease-free equilibrium state and its conditions for the local asymptotic stability.

Control PINN architecture. A visual representation of the Control PINN architecture is found in Fig. 1. Adaptive moment estimation (ADAM) is used as the optimizer. The activation function of exponential linear unit (ELU) is used. The neural density is 100 neurons per layer. There are five hidden layers following the input layer that takes in time (t) and space (x). The information of the system state ($\mathbf{y}$) is passed to the controller ($\mathbf{u}$) both directly and indirectly by a skip connection and three hidden layers, respectively. The aggregate information of the system state ($\mathbf{y}$) and controller ($\mathbf{u}$) is handled similarly in the context of the system's push back on the controller ($\boldsymbol{\lambda}$), except with only two hidden layers. This architecture enables the automatic differentiation of second order and mixed derivatives. This is necessary to impose the custom loss function as detailed earlier in this section.

Choice of optimizer. ADAM is used as the optimizer because at each iteration of training the sampling points are randomized. The resampling of random points is very important to ensure that the PINN finds a nontrivial solution. If static sampling points had been used, then Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) would be appropriate. However, using a fixed set of collocation points can be detrimental to the convergence of

PINNs to the correct solution [10]. Thus, L-BFGS should not be used as the optimizer. That being said, it is worth investigating whether the use of evolutionary sampling (incrementally accumulating collocation points in regions of high PDE residuals) could improve the accuracy of the learned solution by Control PINN [10].

Regularization of weights. The current implementation does not scale boundary conditions, but scales the loss components by a constant (10^{-1}). By regularizing the weights in this way, the scale of each loss component is brought in line to be proportional to one another. In additional experiments not detailed in this paper, constant scaling of the loss components with values less than 10^{-1} resulted in poor performance. Thus, the relative scaling of each loss component's regularized weight was determined using a hand-tuning approach. To justify the choice of constant scaling, and possibly improve upon the accuracy of the learned solutions, future work will include an ablation study of the following: (i) a learning rate annealing algorithm that leverages gradient statistics to balance the interaction between loss components [44]; (ii) curriculum regularization, in which the Control PINN's loss term could start from a simple PDE regularization, and increase in complexity as training progresses [20]; (iii) the augmented Lagrangian relaxation method for PINNs to adaptively balance each loss component [42].

Hyperparameters. In order to minimize the effects of hyperparameters, the network depth, the neural density per layer, and the overall architecture are kept exactly the same for each of the three problems presented in this paper. With all three increasingly challenging experiments presented in this paper using the same architecture, the robustness of the Control PINN framework is demonstrated in its ability to tackle each one without the need for hyperparameter tuning adjustments via random search or grid search.

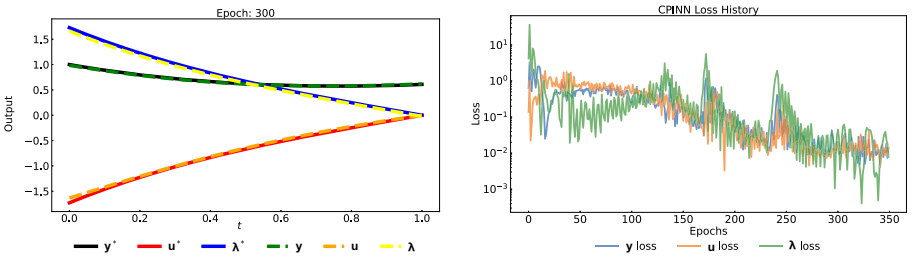
Tension exists between different terms in the loss function defined in Eq. (4). For example, satisfying the boundary conditions may oppose adhering to the constraints imposed by the physical laws. In practice, this is addressed by assigning different weights to the terms in the loss function and treating them as hyperparameters. We further note that as Control PINN is applied to increasingly more complex problems, a unique solution to the problem may not be available. We discuss the validation of the solution found by the Control PINN model as well as its optimality further in Sect. 6.

5 Analytical Problem

As a proof of concept and to provide evidence for the validity of the methodology, we consider the following ODE control problem with a known analytical solution [13, Sect. 6, pp. 272–273, Eq. (65)]:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \Psi(\mathbf{u}, \mathbf{y}) = \arg\min_{\mathbf{u}} \int_{t_0}^{t_f} \left(\mathbf{y}(t)^2 + \frac{1}{2}\mathbf{u}(t)^2 \right) \mathrm{d}t,$$

$$\text{subject to} \quad \begin{cases} \frac{\partial \mathbf{y}}{\partial t}(t) = \frac{1}{2}\mathbf{y} + \mathbf{u}, & \forall t \in [t_0, t_f], \\ \mathbf{y}(t_0) = \mathbf{y}_0 = 1, & \\ \frac{\partial \boldsymbol{\lambda}}{\partial t}(t) = -\frac{1}{2}\boldsymbol{\lambda}(t) - 2\mathbf{y}, & \forall t \in [t_f, t_0], \\ \boldsymbol{\lambda}(t_f) = 0, & \\ 0 = -\boldsymbol{\lambda}(t) - \mathbf{u}, & \forall t \in [t_0, t_f], \end{cases} \tag{5}$$



(a) Learned solution (b) Loss history

Fig. 2 Results for the analytical problem: Eq. (5) **a** the learned solution, control, and adjoint variables match the analytical solution; **b** history of loss components during the training

where $t_0 = 0$ and $t_f = 1$. The optimal solution is known to be

$$\mathbf{y}^*(t) = \frac{2e^{3t} + e^3}{e^{3t/2}(2 + e^3)}, \quad \mathbf{u}^*(t) = \frac{2(e^{3t} - e^3)}{e^{3t/2}(2 + e^3)}, \quad \boldsymbol{\lambda}^*(t) = -\mathbf{u}^*(t). \tag{6}$$

A Control PINN with the architecture described in Sect. 4.1 is trained on this problem. After 300 epochs, the Control PINN converges on the analytical solution, and the solutions remain unchanged upon further training. The results in Fig. 2a, show predicted variables matching the reference solution for various points in time. Figure 2b shows the loss history for different components of the loss function.

6 One-Dimensional Heat Equation

Consider a thin metal beam being heated by a controllable heat source. The objective is to regulate the heat applied so that the beam reaches a specified temperature profile at a desired final time. The governing equation is the one-dimensional unsteady heat equation. The problem is formalized as follows:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \Psi(\mathbf{u}, \mathbf{y}), \tag{7a}$$

$$\begin{aligned} \Psi(\mathbf{u}, \mathbf{y}) = &\int_{\Omega} \left(\mathbf{y}(t_0, x) - \mathbf{y}^*(t_0, x)\right)^2 \, \mathrm{d}x \\ &+ \int_{\Omega} \left(\mathbf{y}(t_f, x) - \mathbf{y}^*(t_f, x)\right)^2 \, \mathrm{d}x \\ &+ \int_{\Omega} \int_{t_0}^{t_f} (\mathbf{u}(t, x))^2 \, \mathrm{d}t \, \mathrm{d}x \end{aligned} \tag{7b}$$

$$\begin{aligned} \text{subject to} \quad & \frac{\partial \mathbf{y}}{\partial t} = 0.1 \frac{\partial^2 \mathbf{y}}{\partial x^2} + \mathbf{u}(t, x), && \forall t \in [t_0, t_f], \ x \in \Omega, \\ & \mathbf{y}(t_0, x) = \sin(\pi x) \sin(2 \pi x), && \forall x \in \Omega, \\ & \mathbf{y}(t, x) = 0, && \forall t \in [t_0, t_f], \ x \in \Omega, \end{aligned} \tag{7c}$$

where $t_0 = 0$, $t_f = 1$, and $\Omega = [0, 1]$. The first term in the objective function in Eq. (7b) ensures that the learned solution obeys the initial conditions, the second term represents the desired final distribution of heat along the beam, and the third term represents the desire

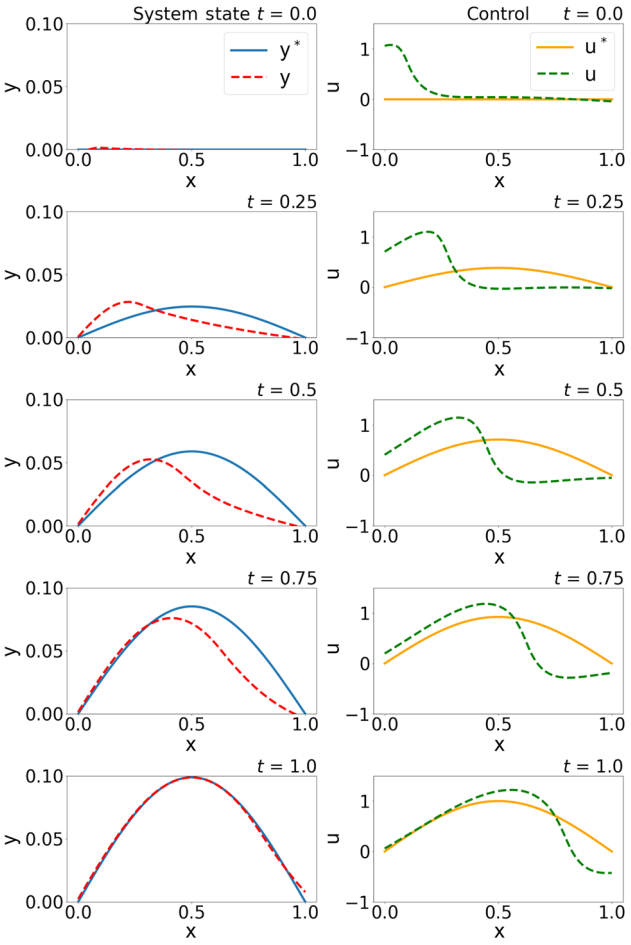
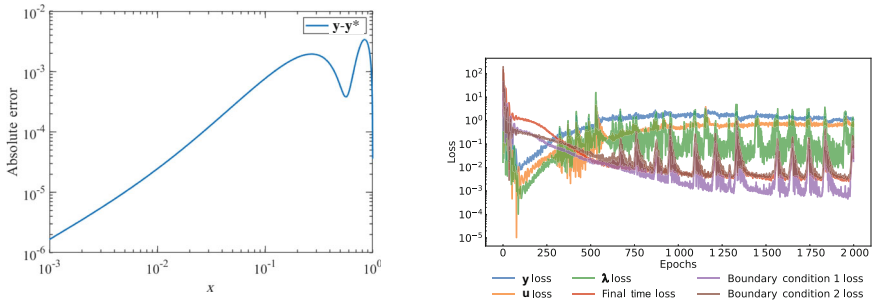


Fig. 3 One-dimensional heat equation results: (left column) comparison of the learned system state $\mathbf{y}$ and the reference system state $\mathbf{y}^*$; (right column) comparison of the learned control $\mathbf{u}$ and the reference control $\mathbf{u}^* = \sin(\pi x)\sin(\frac{\pi t}{2})$. Note that there are multiple control signals that satisfy the PDE solution. The reference control $\mathbf{u}^*$ is sub-optimal, and is not the same as the control signal found by Control PINN. Control PINN found a different control that minimizes the control action on the system state

to minimize the total energy used to control the temperature. We illustrate the results of training the Control PINN model in Fig. 3. As required by Eq. (7), the PINN solution has good agreement with the desired initial and final states. In Fig. 3 we also plot the learned control signal and compare it with the trivial solution $\mathbf{u}^* = \sin(\pi x)\sin(\frac{\pi t}{2})$. Note that there are multiple control signals that satisfy the PDE solution, but they may not be solutions to the optimization problem Eq. (7).

We can further study the accuracy and optimality of the solution found by Control PINN. Since the learned control is a continuous function, it can be sampled at high resolution in time and space. This data, in turn, can be plugged back into an accurate Direct Numerical Simulation (DNS) of the heat equation to compare with the state predictions of the Control PINN. We used the ODE Test Problems package [36] to simulate the system.



(a) Absolute error of the system dynamics (b) History for the boundary conditions

Fig. 4 **a** The absolute error between $\mathbf{y}^*$ and $\mathbf{y}$; **b** Control PINN loss values of the initial, final, and boundary conditions during training

We also take the numerical integral of the squared control signal to approximate the third term in Eq. (7b) for both $\mathbf{u}$ and $\mathbf{u}^*$. Indeed, the numerical validation shows that the Control PINN was able to find a solution with a smaller amount of control (0.247 5 and 0.249 7, respectively).

7 Two-Dimensional Predator-Prey Problem

We next solve a PDE-constrained optimization problem involving the two-dimensional predator-prey (Lotka-Volterra) equation. The dynamics of this problem are formulated as a reaction-diffusion system [21] with the objective:

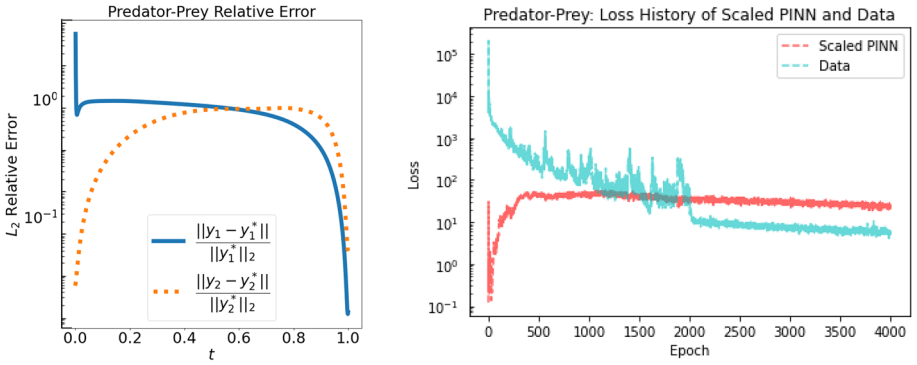
$$\mathbf{u}^* = \arg\min_{\mathbf{u}} \Psi(\mathbf{u}, \mathbf{y}),$$

$$\Psi(\mathbf{u}, \mathbf{y}) = \int_{\Omega}\int_{t_0}^{t_f} \|\mathbf{y}(t,x) - \mathbf{y}^*(t,x)\|_2^2 \,\mathrm{d}t\mathrm{d}x + \int_{\Omega}\int_{t_0}^{t_f} \|\mathbf{u}(t,x)\|_2^2 \,\mathrm{d}t\mathrm{d}x,$$

$$\text{subject to} \quad \begin{cases} 0 = \frac{\partial \mathbf{y}_1}{\partial t} - \frac{\partial^2 \mathbf{y}_1}{\partial x} - u_1(t,x) + \mathbf{y}_1(t,x), & \forall t \in [t_0, t_f], \forall x \in \Omega, \\ 0 = \frac{\partial \mathbf{y}_2}{\partial t} - \frac{\partial^2 \mathbf{y}_2}{\partial x} - u_2(t,x) - \mathbf{y}_2(t,x), & \forall t \in [t_0, t_f], \forall x \in \Omega, \\ \mathbf{y}_1(t_0, x) = \sin(\pi x_1)\sin(\pi x_2), & \forall x \in \Omega, \\ \mathbf{y}_2(t,x) = t\,(\sin(2\pi x_1)\sin(2\pi x_2))^2 + & \\ \qquad (1-t)\sin(\pi x_1)\sin(\pi x_2), & \forall t \in [t_0, t_f], \forall x \in \Omega, \\ \mathbf{y}_1(t,x) = \mathbf{y}_2(t,x) = 0, & \forall x \in \partial\Omega, \end{cases} \tag{8}$$

where $\Omega = [0,1]^2$, $t_0 = 0$, and $t_f = 1$ and the reference state trajectory $\mathbf{y}^*(t,x) = [y_1(t,x), y_2(t,x)]^{\mathrm{T}}$ with

$$\begin{cases} y_1(t,x) &= t \cdot \sin(\pi x_1) \cdot \sin(\pi x_2), \\ y_2(t,x) &= (1-t) \cdot \sin(\pi x_1) \cdot \sin(\pi x_2) + t \cdot (\sin(2\pi x_1) \cdot \sin(2\pi x_2))^2 . \end{cases} \tag{9}$$



(a) *Relative error* (b) *PINN vs. Data loss*

Fig. 5 The error and loss history for the two-dimensional predator-prey problem: **a** L_2 norm of the relative error over the physical domain as a function of time; **b** history of the scaled PINN and data loss during training

The system characterized by Eq. (8) defines the interaction of two populations: the predator and the prey. The birth rate for each population follows an exponential law based on the current population and is also affected by the competing population.

We are interested in controlling the prey population by introducing the predator population at different times and locations. For this example, we have considered a starting profile of the prey population, which is then "herded" into a desired profile over a specified timespan. The predator population in Eq. (8) is denoted by $\mathbf{y}_1$, and $\mathbf{y}_2$ denotes the prey population. Similarly, u_1 and u_2 are predator and prey control functions, respectively. As we only apply control to the prey population in this problem, we set $u_1(t, x) = 0$.

Once again, we train the model described in Sect. 4.1 on this problem. The constraints are on both populations due to boundary and initial conditions, as well as the entire trajectory of the state as described in Eq. (9). In addition to that, we would like to minimize the amount of control applied (see Ψ in Eq. (1)).

Figure 6 shows the reference state variables at the final time compared to the Control PINN states, verifying small errors in the learned state solutions. We also plot the norm of absolute error as a function of time in Fig. 5a. We see the objective is satisfied well at the initial and final times, while there is some mismatch at other time points. Figure 5b shows the interaction of two major loss terms during training: the reduction in data loss shows an improving fit to the training data while the PINN loss is kept relatively comparable to guarantee near optimality of the learned control and boundary and initial condition satisfaction.

Since this problem does not have an analytical control solution, there is no direct comparison that we can make for the learned control. However, interested readers may inspect the evolution of the learned system states and the control signal from $t = 0$ to $t = 1$ in "Appendix C".

8 Conclusions

This work provides a novel approach for solving the optimal control problem for PDEs using PINNs. In contrast to previous approaches, we integrate the optimality conditions from

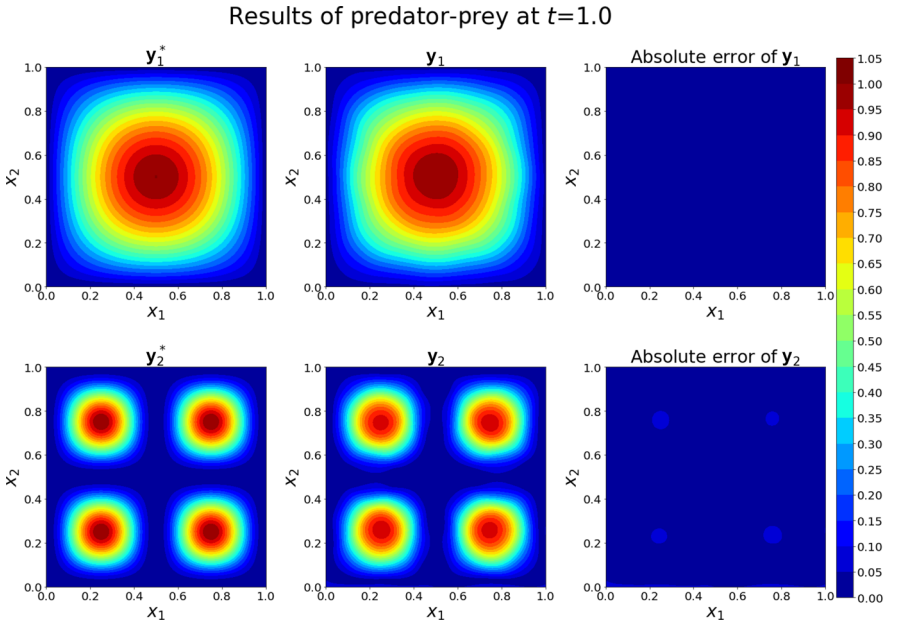


Fig. 6 Two-dimensional predator-prey results: comparison of both the reference solution and the learned solution of the predator and prey populations at the final time $t = 1.0$. The reference solution of the predator population is denoted by $\mathbf{y}_1^*$. The learned solution of the predator population is denoted by $\mathbf{y}_1$. The learned solution of the prey population is denoted by $\mathbf{y}_2$. The reference solution of the prey population is denoted by $\mathbf{y}_2^*$. The right-most column displays the absolute error between the reference and learned solutions for each population. An expanded looked into the absolute errors throughout the timespan $t = [0, 1]$ is found in "Appendix C"

the control problem directly in a theory-guided and physics-informed manner. We dub this approach Control PINNs.

The Control PINN methodology is able to simultaneously learn the system state solution and the optimal control for a general class of PDEs. We illustrate the validity of our approach on a diverse set of problems: a simple control problem for which an analytical solution is known, a one-dimensional heat equation for which a control that is not optimal is known, and a two-dimensional predator-prey problem which might not even have an optimal control. In higher dimensional problems, a tension exists in Control PINNs between respecting the boundary conditions of the given system state while learning the solution and corresponding optimal control. Adaptive methods for choosing the scaling of the terms in the loss similar to [45] are of future interest.

Future work will involve extending control PINNs to solve closed-loop control problems. [12, 23, 29] offer PINN approaches with which to compare and benchmark Control PINN's performance. Moreover, more complicated experiments such as Navier-Stokes will be utilized in subsequent complementary papers. Future results will be sure to offer a comparison between the adjoint and the PINN with no adjoint. The computational costs, modes of failure, and scope of achievement will accompany said results. After establishing Control PINN's viability in closed-loop control problems, one potential future application is leveraging Control PINNs as agents in deep reinforcement learning (DRL) [9]. In DRL, finding the state of a robot after a given action requires solving a number of physical equations (e.g., equation of motion and balance of force). This issue can be circumvented by leveraging PINNs as an

agent because PINNs penalize deviations from physical constraints by design. Furthermore, an agent that can simultaneously solve a system state and a corresponding optimal control, such as Control PINN, would be useful in Q-learning to efficiently optimize the value of action-state policies.

A more efficient way to model and optimally control machines interacting with the real world has various societal impacts. Faster modeling begets faster understanding, and consequently a positive feedback loop can be established regarding answering certain questions of interest. With regard to potential negative impacts on society, offloading multistage decision processes to machines in the form of an optimal control policy lends credence to the idea of a future reduction in the demand for human labor. Energy resource optimization, industrial automation, and urban complex systems could be affected by combining optimal control and machine learning in the form of Control PINNs.

Compute resources. The GitHub repository for this paper is at https://github.com/ComputationalScienceLaboratory/control-pinns, and contains Google Colaboratory (Colab) notebooks for each experiment that are self-contained. The Colab notebook associated with the predator-prey problem in Sect. 7 takes approximately forty-five minutes to run using a Tesla P100-PCIE-16GB GPU, uses 2.30 GB of RAM, and 38.74 GB of disk space. The other experiments are considerably faster to run, and less demanding resource-wise.

Appendix A Analytical Problem

See Figs. A1 and A2.

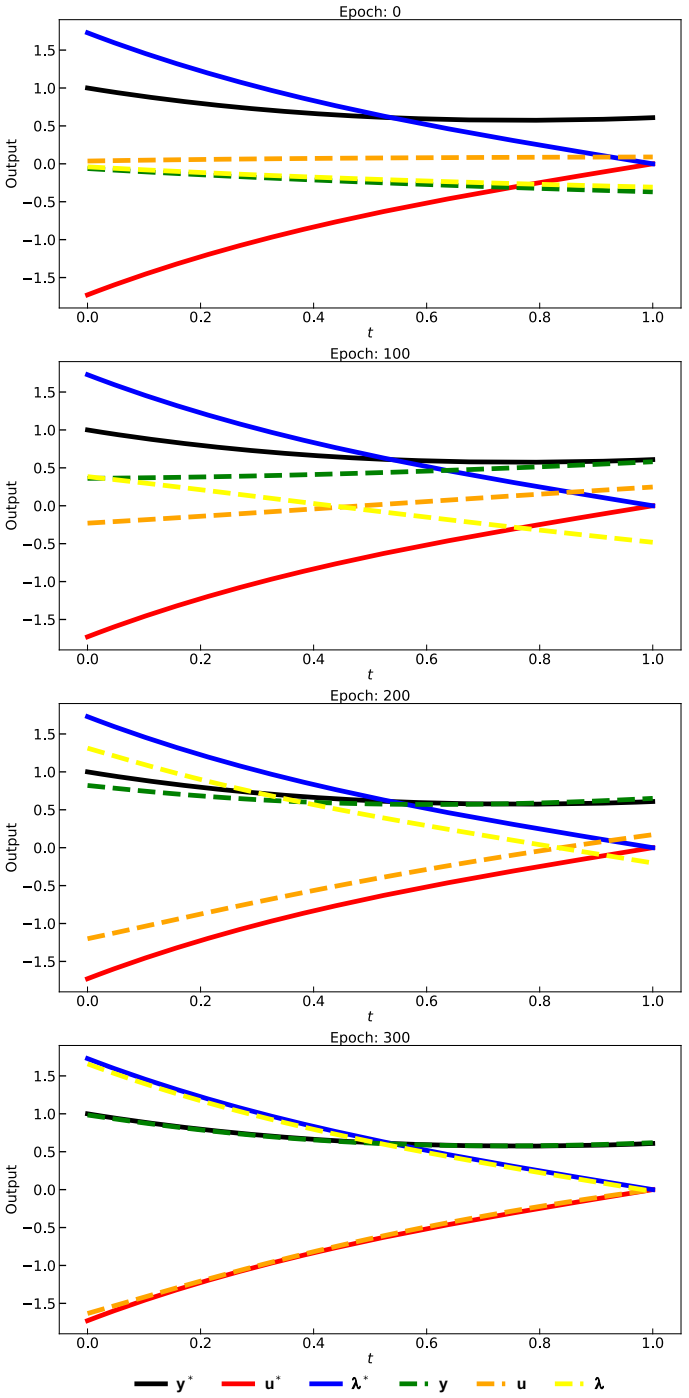


Fig. A1 Results of the analytical problem: the optimal solutions of the system state, control, and adjoint on the control are respectively denoted by $\mathbf{y}^*$, $\mathbf{u}^*$, and $\boldsymbol{\lambda}^*$. Their corresponding learned solutions found by the Control PINN model are denoted by $\mathbf{y}$, $\mathbf{u}$, and $\boldsymbol{\lambda}$. After 300 epochs Control PINN has reached convergence on the analytical solution, and the solutions remain unchanged upon further training

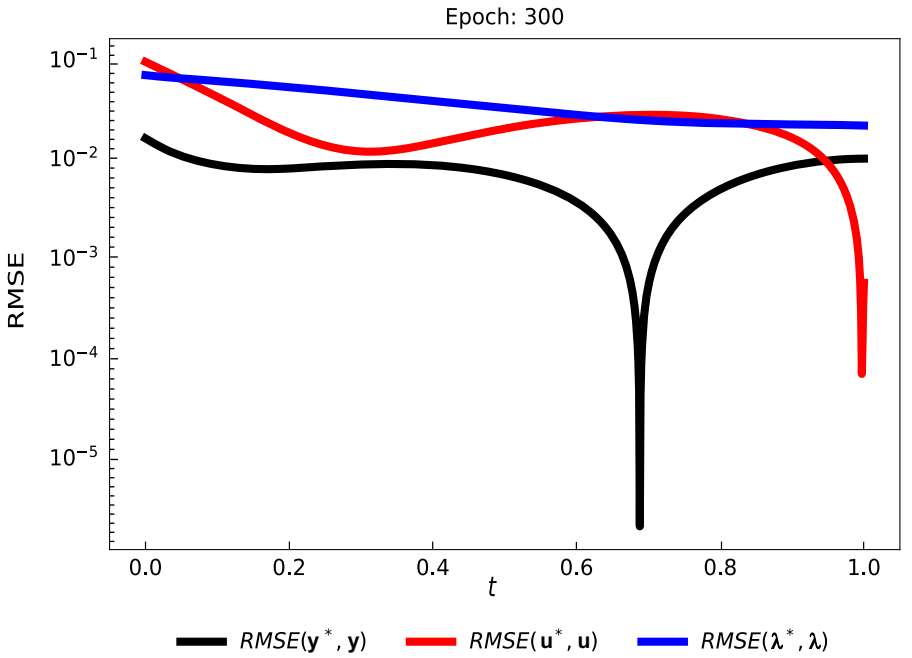


Fig. A2 Results of the analytical problem: the root mean squared error (RMSE) between the optimal solutions of the system state, control, and adjoint on the control ($\mathbf{y}$, $\mathbf{u}$, and $\boldsymbol{\lambda}$) and their respective analytical solutions ($\mathbf{y}^*$, $\mathbf{u}^*$, and $\boldsymbol{\lambda}^*$)

Appendix B One-Dimensional Heat Equation

See Figs. B1 and B2.

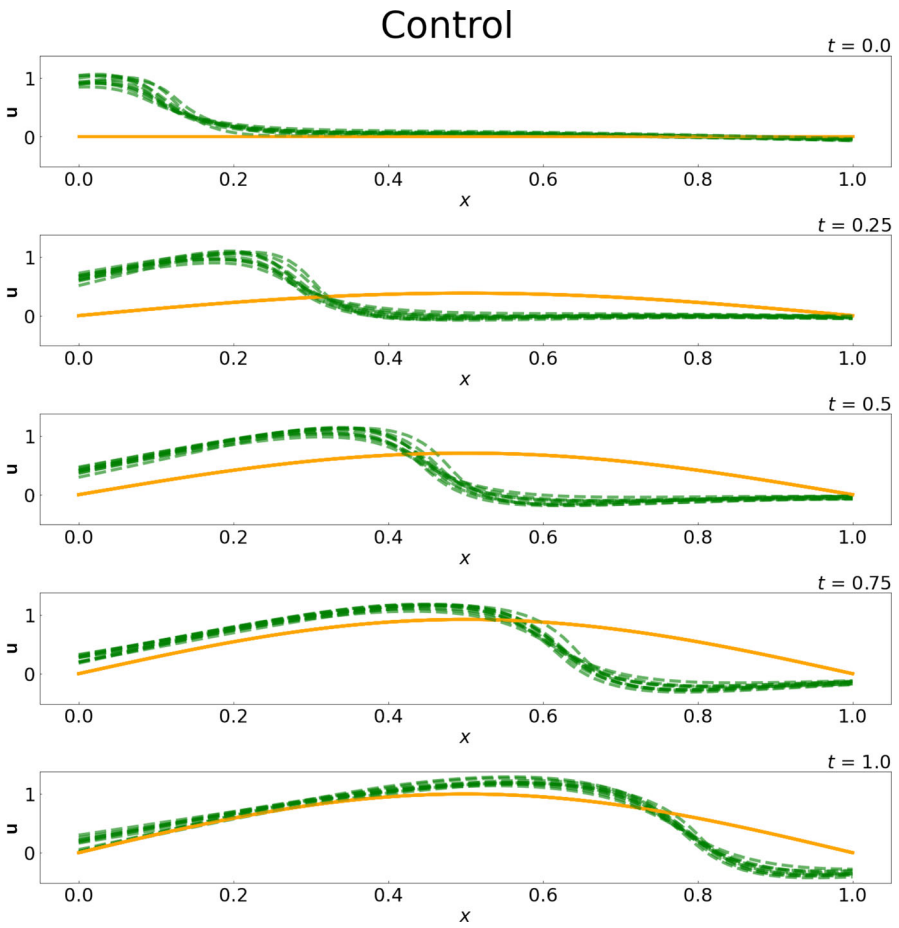


Fig. B1 Stability analysis of control signal: the Control PINN architecture was trained on ten different random initialization of weights, resulting in learning the same approximate control signal solution for the one-dimensional heat equation in Sect. 6. The solid yellow line represents the reference control signal $\mathbf{u}^*$, whereas the dotted green lines represent the ten optimal control signals $\mathbf{u}$ learned by the Control PINN framework

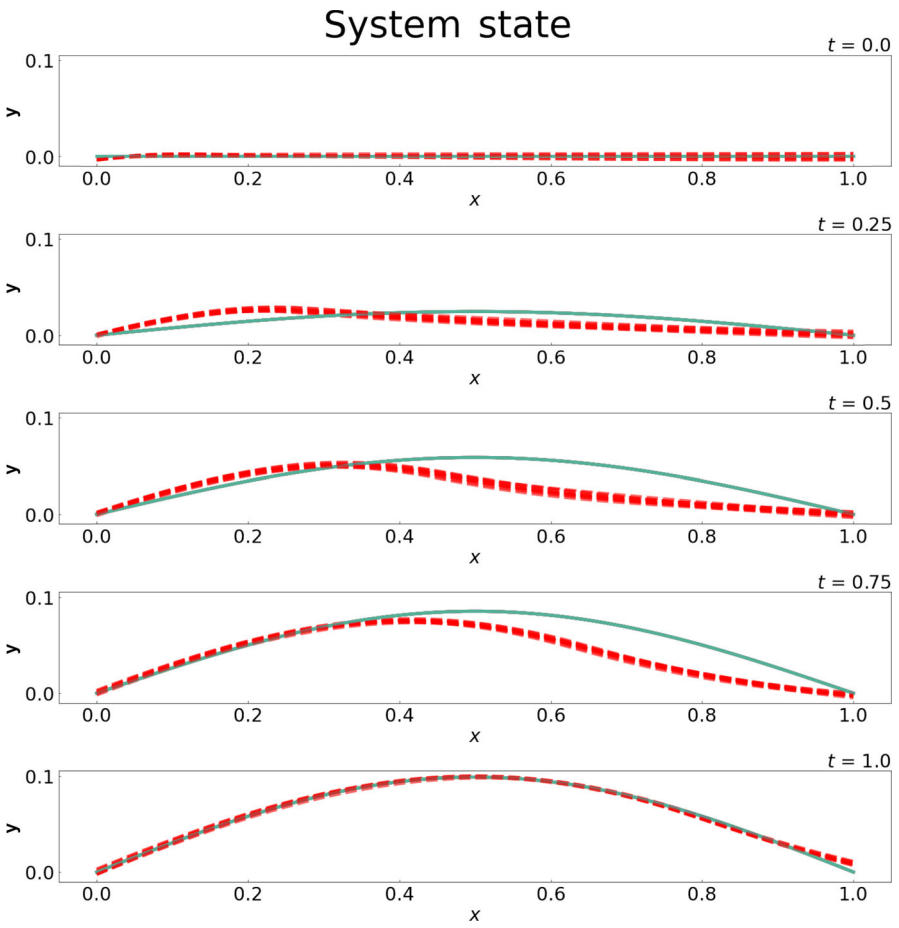


Fig. B2 Stability analysis of system dynamics: the Control PINN architecture was trained on ten different random initialization of weights, resulting in learning the same approximate system dynamics for the one-dimensional heat equation in Sect. 6. The solid blue line represents the reference system dynamics $\mathbf{y}^*$, whereas the dotted red lines represent the ten optimal system dynamics $\mathbf{y}$ learned by the Control PINN framework

Appendix C Two-Dimensional Predator-Prey Problem

See Figs. C1, C2, and C3.

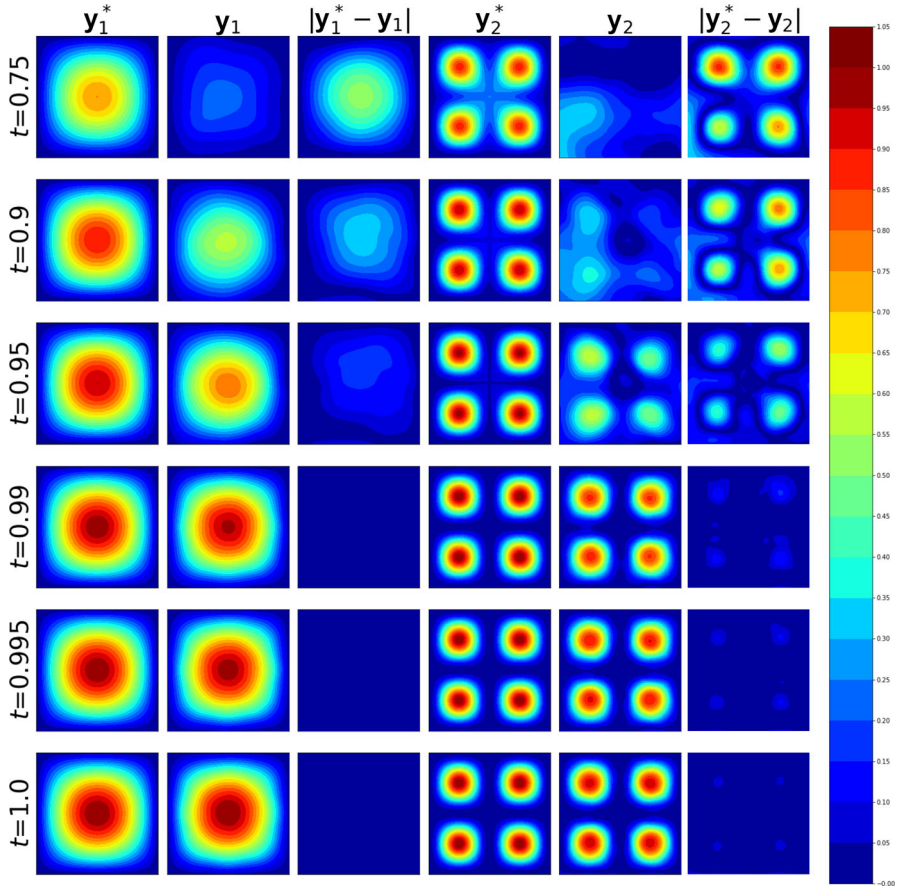


Fig. C1 Two-dimensional predator-prey results: expanded comparison of the absolute error between the reference and learned solutions for the timepoints $t \in [0.75, 0.9, 0.95, 0.99, 0.995, 1.0]$, with each row in the plot denoting a given point in the timespan. (First column) The reference solution $\mathbf{y}_1^*$ of the system, and its corresponding state respective to the given timepoint row. (Second column) The learned solution $\mathbf{y}_1$ of the system by Control PINN, and its corresponding state respective to the given timepoint row. (Third column) The absolute error between the reference and learned solutions of the predator populations ($\mathbf{y}_1^*$ and $\mathbf{y}_1$, respectively). (Fourth column) The reference solution $\mathbf{y}_2^*$ of the system, and its corresponding state respective to the given timepoint row. (Fifth column) The learned solution $\mathbf{y}_2$ of the system by Control PINN, and its corresponding state respective to the given timepoint row. (Sixth column) The absolute error between the reference and learned solutions of the prey populations ($\mathbf{y}_2^*$ and $\mathbf{y}_2$, respectively)

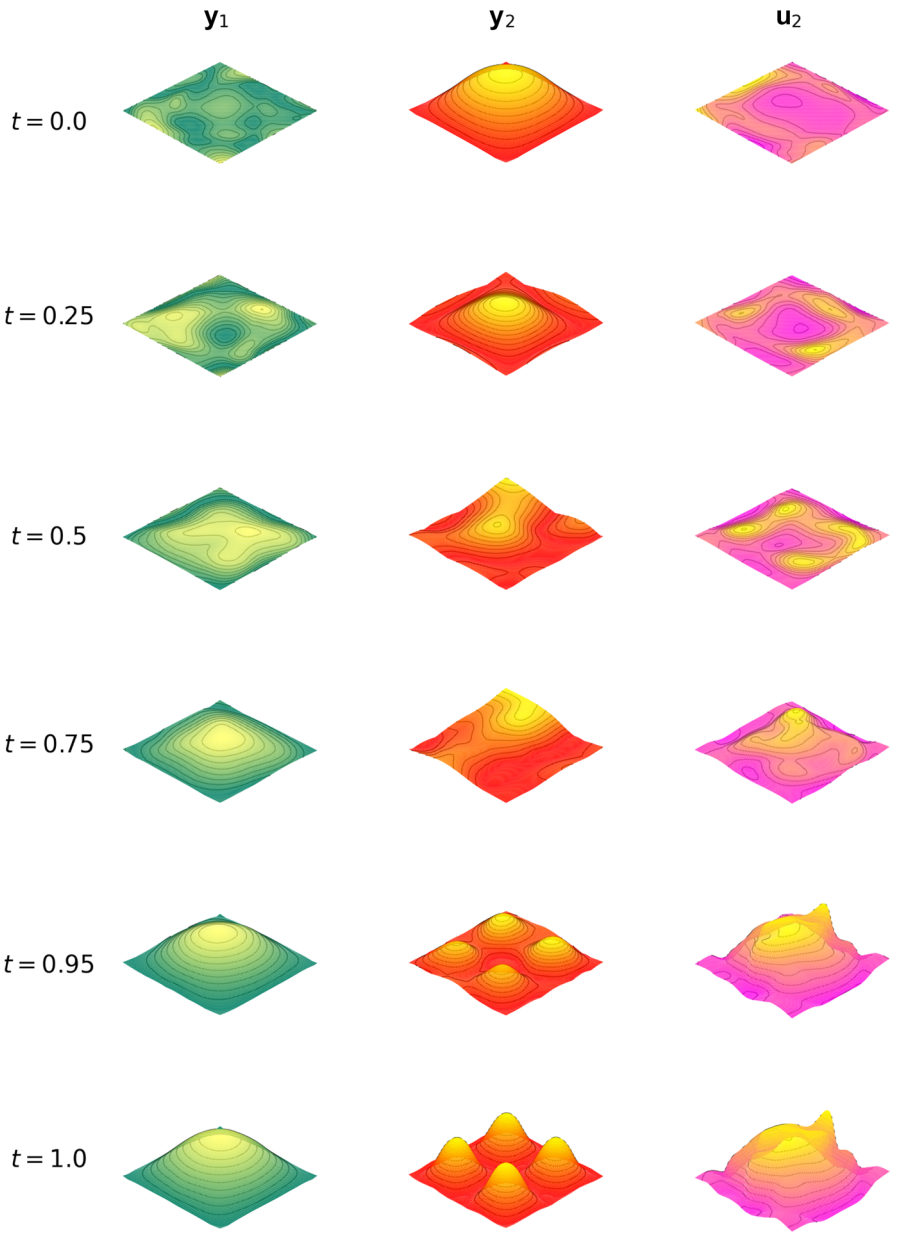


Fig. C2 Two-dimensional predator-prey problem: surface plots of the learned solutions for the predator population, prey population, and control at varying time points. (First column) The learned solution of the predator population by Control PINN, denoted by $\mathbf{y}_1$. (Second column) The learned solution of the prey population by Control PINN, denoted by $\mathbf{y}_2$. (Third column) The learned solution of the control signal by Control PINN, denoted by $\mathbf{u}_2$. Control PINN minimizes the control needed to reach the desired population states by waiting until near the end to enact significant control on the prey population

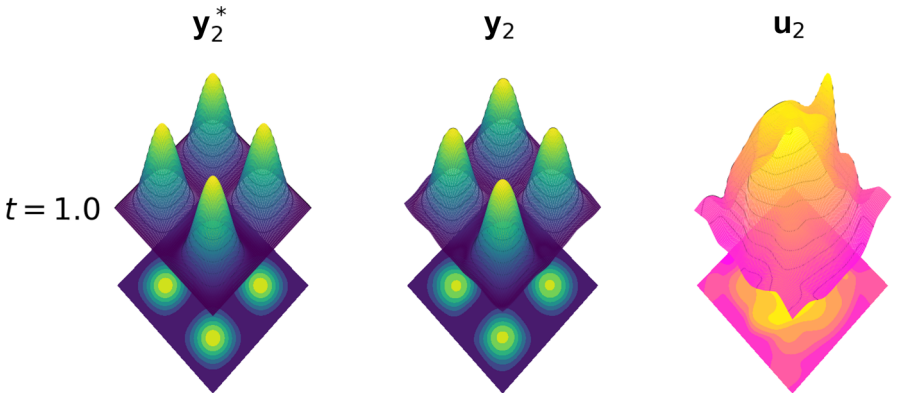


Fig. C3 Two-dimensional predator-prey problem. (Left) Prescribed prey population at the final time $t_f = 1.0$. (Middle) Learned solution $\mathbf{y}_2$ prey population at the final time $t_f = 1.0$. (Right) Control signal $\mathbf{u}_2$ on the predator population $\mathbf{y}_2$ determined by the Control PINN at the final time $t_f = 1.0$. The three-dimensional surface plot is projected downward to a two-dimensional contour plot for increased discrepancy of comparative purposes. Note the differences in the bases of the surface plots, which illustrate the challenges of boundary condition enforcement

Acknowledgements The authors would like to thank Austin Chennault for pointing out many relevant pieces of literature to this research.

Funding This work was supported in part by awards, DOE ASCR DE-SC0021313, NSF DMS-2411069, NSF DMS-2436357, and by the Computational Science Laboratory at Virginia Tech. Arash Sarshar's work used Jetstream2 at Indiana University through allocation CIS230277 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by the National Science Foundation (Grant Nos. 2138259, 2138286, 2138307, 2137603, and 2138296).

Data Availability The code for this paper is available at https://github.com/ComputationalScienceLaboratory/control-pinns.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical Approval Not applicable.

References

1. Antonelo, E.A., Camponogara, E., Seman, L.O., Jordanou, J.P., de Souza, E.R., Hübner, J.F.: Physics-informed neural nets for control of dynamical systems. Neurocomputing **579**, 127419 (2024)
2. Atzberger, P. J. Importance of the mathematical foundations of machine learning methods for scientific and engineering applications. arXiv:1808.02213 (2018)
3. Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966). (**Accessed 2022-05-15**)

4. Biegler, L. T., Ghattas, O., Heinkenschloss, M., van Bloemen Waanders, B. Large-scale PDE-constrained optimization: an introduction. In: Lecture Notes in Computational Science and Engineering, vol. 30. Springer, Berlin/Heidelberg (2003)
5. Blonigan, P.J., Fernandez, P., Murman, S.M., Wang, Q., Rigas, G., Magri, L.: Toward a chaotic adjoint for LES (2017). https://doi.org/10.48550/ARXIV.1702.06809
6. Chen, Y., Lu, L., Karniadakis, G.E., Dal Negro, L.: Physics-informed neural networks for inverse problems in nano-optics and metamaterials. Opt. Express **28**(8), 11618 (2020). https://doi.org/10.1364/oe.384875
7. Chen, Y., Shi, Y., Zhang, B.: Optimal control via neural networks: a convex approach. arXiv:1805.11835 (2018)
8. Chennault, A., Popov, A.A., Subrahmanya, A.N., Cooper, R., Karpatne, A., Sandu, A.: Adjoint-matching neural network surrogates for fast 4D-Var data assimilation (2021). https://doi.org/10.48550/ARXIV.2111.08626
9. Cuomo, S., Di Cola, V.S., Giampaolo, F., Rozza, G., Raissi, M., Piccialli, F.: Scientific machine learning through physics-informed neural networks: where we are and what's next. J. Sci. Comput. **92**(3), 88 (2022)
10. Daw, A., Bu, J., Wang, S., Perdikaris, P., Karpatne, A.: Rethinking the importance of sampling in physics-informed neural networks (2022). https://doi.org/10.48550/ARXIV.2207.02338
11. De Florio, M., Schiassi, E., Calabrò, F., Furfaro, R.: Physics-informed neural networks for 2nd order odes with sharp gradients. J. Comput. Appl. Math. **436**, 115396 (2024)
12. Gokhale, G., Claessens, B., Develder, C.: Physics informed neural networks for control oriented thermal modeling of buildings. Appl. Energy **314**, 118852 (2022). https://doi.org/10.1016/j.apenergy.2022.118852
13. Hager, W.W.: Runge-Kutta methods in optimal control and the transformed adjoint system. Numer. Math. **87**, 247–282 (2000)
14. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems. Springer, Berlin (2002)
15. He, Q., Barajas-Solano, D., Tartakovsky, G., Tartakovsky, A.M.: Physics-informed neural networks for multiphysics data assimilation with application to subsurface transport. Adv. Water Resour. **141**, 103610 (2020). https://doi.org/10.1016/j.advwatres.2020.103610
16. Hinze, M., Pinnau, R., Ulbrich, M., Ulbrich, S.: Optimization with PDE Constraints. Springer, Dordrecht, Netherlands (2008). https://doi.org/10.1007/978-1-4020-8839-1
17. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5), 359–366 (1989)
18. Hwang, R., Lee, J.Y., Shin, J.Y., Hwang, H.J.: Solving PDE-constrained control problems using operator learning. Proceedings of the AAAI Conference on Artificial Intelligence **36**(4), 4504–4512 (2022)
19. Kollmannsberger, S., D'Angella, D., Jokeit, M., Herrmann, L.: Physics-Informed Neural Networks, pp. 55–84. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-76587-3-5
20. Krishnapriyan, A.S., Gholami, A., Zhe, S., Kirby, R.M., Mahoney, M.W.: Characterizing possible failure modes in physics-informed neural networks (2021). https://doi.org/10.48550/ARXIV.2109.01050
21. Lotka, A.J.: Contribution to the theory of periodic reactions. J. Phys. Chem. **14**(3), 271–274 (1910). https://doi.org/10.1021/j150111a004
22. Lu, L., Meng, X., Mao, Z., Karniadakis, G.: DeepXDE: a deep learning library for solving differential equations. SIAM Rev. **63**(1), 208–228 (2021). https://doi.org/10.1137/19M1274067
23. Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., Johnson, S.G.: Physics-informed neural networks with hard constraints for inverse design (2021). https://doi.org/10.48550/ARXIV.2102.04626
24. Mowlavi, S., Nabi, S.: Optimal control of PDEs using physics-informed neural networks. J. Comput. Phy. **473**, 111731 (2023)
25. Mrope, F., Kigodi, O.: Modeling the transmission dynamics of maize foliar disease in maize plants: modeling maize foliar disease dynamics in maize plants. J. Math. Anal. Model. **5**(2), 114–135 (2024)
26. Mrope, F., Nyerere, N.: Modeling the transmission dynamics of bushfires on cashew nut production. J. Math. Anal. Model. **5**(2), 98–113 (2024). https://doi.org/10.48185/jmam.v5i2.1183
27. Nabian, M.A., Meidani, H.: A deep learning solution approach for high-dimensional random differential equations. Probab. Eng. Mech. **57**, 14–25 (2019). https://doi.org/10.1016/j.probengmech.2019.05.001
28. Nellikkath, R., Chatzivasileiadis, S.: Physics-informed neural networks for minimising worst-case violations in DC optimal power flow. In: IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), pp. 419–424 (2021)
29. Nicodemus, J., Kneifl, J., Fehr, J., Unger, B.: Physics-informed neural networks-based model predictive control for multi-link manipulators (2021). https://doi.org/10.48550/ARXIV.2109.10793
30. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (1999)

31. Pang, G., Lu, L., Karniadakis, G.E.: fPINNs: fractional physics-informed neural networks. SIAM J. Sci. Comput. **41**(4), 2603–2626 (2019). https://doi.org/10.1137/18m1229845
32. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations. arXiv:1711.10561 (2017a)
33. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics informed deep learning (part II): data-driven discovery of nonlinear partial differential equations. arXiv:1711.10566 (2017b)
34. 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. **378**, 686–707 (2019)
35. Raissi, M., Yazdani, A., Karniadakis, G.: Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations. Science **367**, 4741 (2020). https://doi.org/10.1126/science.aaw4741
36. Roberts, S., Popov, A.A., Sandu, A.: ODE test problems: a MATLAB suite of initial value problems. arXiv:1901.04098 (2019)
37. Roberts, S., Popov, A.A., Sarshar, A., Sandu, A.: A fast time-stepping strategy for dynamical systems equipped with a surrogate model. SIAM J. Sci. Comput. **44**(3), 1405–1427 (2022). https://doi.org/10.1137/20M1386281
38. Shafiullah, K.: Existence theory and stability analysis to a class of hybrid differential equations using confirmable fractal fractional derivative. J. Fract. Calc. Nonlinear Syst. **7**(8) (2024)
39. Shah, K., Abdeljawad, T., Alrabaiah, H.: On coupled system of drug therapy via piecewise equations. Fractals **30**(08), 2240206 (2022)
40. Shah, K., Sarwar, M., Abdeljawad, T.: On mathematical model of infectious disease by using fractals fractional analysis. Discr. Cont. Dyn. Syst. **17**(10), 3064–3085 (2024)
41. Shah, K., Sinan, M., Abdeljawad, T., El-Shorbagy, M.A., Abdalla, B., Abualrub, M.S.: A detailed study of a fractal-fractional transmission dynamical model of viral infectious disease with vaccination. Complexity **2022**(1), 7236824 (2022)
42. Son, H., Cho, S.W., Hwang, H.J.: AL-PINNs: augmented Lagrangian relaxation method for physics-informed neural networks (2022). https://doi.org/10.48550/ARXIV.2205.01059
43. Wang, S., Bhouri, M.A., Perdikaris, P.: Fast PDE-constrained optimization via self-supervised operator learning. arXiv:2110.13297 (2021)
44. Wang, S., Teng, Y., Perdikaris, P.: Understanding and mitigating gradient pathologies in physics-informed neural networks (2020). https://doi.org/10.48550/ARXIV.2001.04536
45. Wang, S., Wang, H., Perdikaris, P.: Improved architectures and training algorithms for deep operator networks (2021). https://doi.org/10.48550/ARXIV.2110.01654
46. Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V.: Integrating scientific knowledge with machine learning for engineering and environmental systems ACM Comput. Surv. **55**(4), 1–37, 66 (2023)
47. Yan, X.-B., Xu, Z.-Q.J., Ma, Z.: Bayesian Inversion with Neural Operator (BINO) for modeling subdiffusion: forward and inverse problems. J. Comput. Appl. Math. **454**, 116191 (2025). https://doi.org/10.1016/j.cam.2024.116191
48. Yang, L., Zhang, D., Karniadakis, G.E.: Physics-informed generative adversarial networks for stochastic differential equations. SIAM J. Sci. Comp. **42**(1), A292–A317 (2020)
49. Yu, T.F., Chung, E.T., Cheung, K.C., Zhao, L.: Learning computational upscaling models for a class of convection-diffusion equations. J. Comput. Appl. Math. **445**, 115814 (2024). https://doi.org/10.1016/j.cam.2024.115814
50. Zhang, D., Guo, L., Karniadakis, G.E.: Learning in modal space: solving time-dependent stochastic PDEs using physics-informed neural networks. SIAM J. Sci. Comp. **42**(2), A639–A665 (2020)
51. Zhang, D., Lu, L., Guo, L., Karniadakis, G.E.: Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. J. Comput. Phys. **397**, 108850 (2019). https://doi.org/10.1016/j.jcp.2019.07.048