Safe PDE Boundary Control with Neural Operators

Hanjiang Hu HANJIANGHU@CMU.EDU and **Changliu Liu** CLIU6@ANDREW.CMU.EDU
The Robotics Institute, Carnegie Mellon University

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

The physical world dynamics are generally governed by underlying partial differential equations (PDEs) with unknown analytical forms in science and engineering problems. Neural network based data-driven approaches have been heavily studied in simulating and solving PDE problems in recent years, but it is still challenging to move forward from understanding to controlling the unknown PDE dynamics. PDE boundary control instantiates a simplified but important problem by only focusing on PDE boundary conditions as the control input and output. However, current model-free PDE controllers cannot ensure the boundary output satisfies some given user-specified safety constraint. To this end, we propose a safety filtering framework to guarantee the boundary output stays within the safe set for current model-free controllers. Specifically, we first introduce a neural boundary control barrier function (BCBF) to ensure the feasibility of the trajectory-wise constraint satisfaction of boundary output. Based on the neural operator modeling the transfer function from boundary control input to output trajectories, we show that the change in the BCBF depends linearly on the change in input boundary, so quadratic programming-based safety filtering can be done for pre-trained model-free controllers. Extensive experiments under challenging hyperbolic, parabolic and Navier-Stokes PDE dynamics environments validate the plug-and-play effectiveness of the proposed method by achieving better general performance and boundary constraint satisfaction compared to the vanilla and constrained model-free controller baselines. The code is available at https://github.com/intelligent-control-lab/safe-pde-control.

Keywords: PDE control, safety filter, learning for control

1. Introduction

Partial differential equations (PDEs) characterize the most fundamental laws of the continuous dynamical systems in the physical world (Evans, 1998; Perko, 1996). Non-analytical PDE dynamics are often involved in complicated science and engineering problems of computational fluid dynamics (Kochkov et al., 2021), computational mechanics (Samaniego et al., 2020), robotics (Heiden et al., 2021), etc. Recently, neural networks have largely boosted the study of numerical PDE solvers using data-driven methods, simulating and characterizing the dynamics (Raissi et al., 2019; Brunton and Kutz, 2024; Kovachki et al., 2023). However, the PDE control problem remains challenging without any prior knowledge about underlying PDE equations, serving as a huge gap from understanding science to solving engineering problems (Yu and Wang, 2024).

Recent pioneer works (Bhan et al., 2024; Zhang et al., 2024a) provide various formulations of PDE control problems and multiple benchmark settings, either in-domain control (Zhang et al., 2024b) or boundary control (Bhan et al., 2023). Since it is easier to control the PDE boundary in the real world, following Bhan et al. (2024), we focus on the PDE boundary control setting where the control signal essentially serves as the boundary condition and the unknown PDE dynamics itself remains unchanged. Model-based PDE boundary control has been studied for years, and backstepping-based methods have been applied to different PDE dynamics (Krstic and Smyshlyaev,

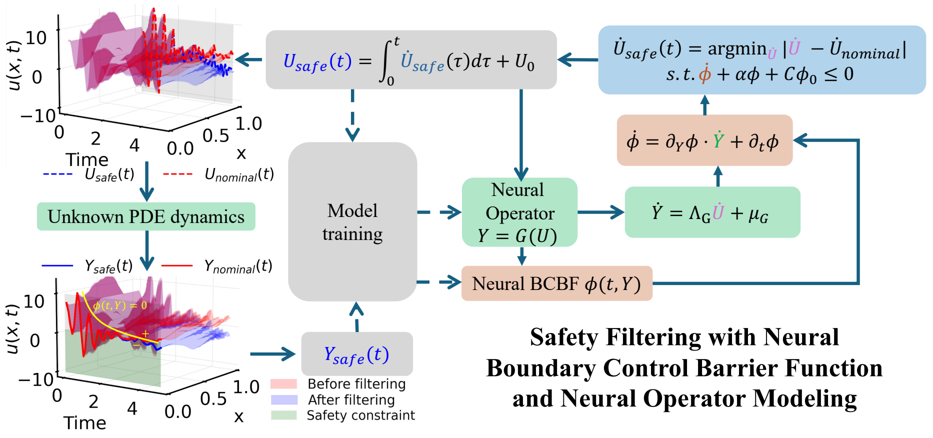
$U_{safe}(t) = \int_0^t \dot{U}_{safe}(\tau)d\tau + U_0$
$\dot{U}_{safe}(t) = \text{argmin}_{\dot{U}} |\dot{U} - \dot{U}_{nominal}|$
$s.t. \dot{\phi} + \alpha\phi + C\phi_0 \leq 0$
$\dot{\phi} = \partial_Y \phi \cdot \dot{Y} + \partial_t \phi$
$\dot{Y} = \Lambda_G \dot{U} + \mu_G$
Neural Operator $Y = G(U)$
Neural BCBF $\phi(t, Y)$
Model training
Unknown PDE dynamics
$U_{safe}(t)$
$U_{nominal}(t)$
$Y_{safe}(t)$
$Y_{nominal}(t)$
Before filtering
After filtering
Safety constraint
Safety Filtering with Neural Boundary Control Barrier Function and Neural Operator Modeling

Figure 1: Overview of our safety filtering method for PDE boundary control with neural BCBF. Solid line arrows denote the safety filtering, while dashed ones denote the model training.

2008). Nevertheless, the model-based methods cannot work well under the unknown PDE dynamics, suffering from significant model mismatch. Model-free reinforcement learning (RL) controllers (Schulman et al., 2017; Haarnoja et al., 2018) have shown impressive results in the benchmark (Bhan et al., 2024) compared to the model-based control methods (Pyta et al., 2015).

Besides, constraint satisfaction is of great importance for the PDE boundary control problems, but current safe PDE control methods are typically backstepping-based and require knowledge about the PDE dynamics (Krstic and Bement, 2006; Li and Krstic, 2020; Koga and Krstic, 2023; Wang and Krstic, 2023). The constraint considered in this paper is called *boundary feasibility*, which characterizes whether the boundary output falls into and stays within the safe set at the end of the finite-time trajectory, and can be understood as the constraint of finite-time convergence. Under ordinary differential equations (ODEs) setting, neural network parameterized control Lyapunov/barrier functions (CLF/CBFs) have been adopted to ensure the convergence and safety of learning-based controllers (Boffi et al., 2021; Dawson et al., 2023; Chang et al., 2019; Mazouz et al., 2022), based on the Markov property of the dynamics at each step , i.e., the change of state only depends on the current state and control input. However, the Markov assumption does not generally hold for PDE boundary control due to infinite-dimensional unobserved states along the spatial axis. It is also challenging to bypass the unknown PDE dynamics to to find the boundary control input at each step for trajectory-wise convergence over boundary output constraint.

To this end, we introduce a new framework to achieve *boundary feasibility* within a given safe set for the PDE boundary control problem, as shown in Figure 1. More specifically, we propose neural boundary control barrier functions (BCBFs) over the boundary output to enable the incorporation of the time variable with a finite-time convergence guarantee. Then, we adopt a neural operator to directly learn the mapping from boundary input to output as a transfer function. Combining well-trained neural BCBF and neural operator, we show a linear dependence between boundary feasibility condition and the derivative of boundary control input, making the safety filtering possible by projecting the actions from the nominal RL controller to the safe boundary control input set using quadratic programming (QP). We conduct experiments on multiple PDE benchmarks and show our plug-and-play filtering superiority over vanilla and constrained RL controllers regarding general performance and constraint satisfaction. To the best of our knowledge, we are the first to study safe boundary control with unknown PDE dynamics. More related work is discussed in Section A in Hu and Liu (2025). We summarize our contributions below.

- We propose a new PDE safe control framework with a neural boundary control barrier function to guarantee the boundary feasibility of the boundary output within a given safe set.
- We model the control input and output mapping through a neural operator as a transfer function and prove that it can be used for safety filtering by solving quadratic programming.
- We show that the add-on performance after safety filtering is better than both vanilla and constrained RL controllers in boundary feasibility rate and time steps on multiple PDE environments.

2. Problem Formulation

Following the PDE boundary control setting (Bhan et al., 2024), we consider the state $u(x,t) : \mathcal{X} \times \mathcal{T} \to \mathcal{S} \subset \mathbb{R}$ from the continuous function space $C(\mathcal{X} \times \mathcal{T}; \mathbb{R})$ governed by underlying closed-loop partial differential equation (PDE) dynamics defined on normalized n-dimensional spatial domain $\mathcal{X} = [\mathbf{0}, \mathbf{1}] := [0,1]^n \subset \mathbb{R}^n$ and temporal domain $\mathcal{T} = [0,T] \subset \mathbb{R}^+$ as follows,

$$\frac{\partial u}{\partial t} = \mathcal{D}(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \dots, U(t)), x \in \mathcal{X}, t \in \mathcal{T}, u \in \mathcal{S}, \tag{1}$$

where $\mathcal{D}$ is the PDE system dynamics and $U(t)$ is the control signal as the boundary condition. Without loss of generality, we focus on the Dirichlet boundary control input as $U(t) := u(\mathbf{1}, t)$ with constant initial condition $u(x,0) \equiv U(0) \in \mathcal{S}$. Instead of optimizing boundary input $U(t)$ to track or stabilize full-state observation trajectory $u(x,t)$ (Bhan et al., 2024), we aim to find $U(t)$ that guarantees the *boundary feasibility* of boundary output $Y(t) := u(\mathbf{0}, t)$ within the given user-specified safe set $\mathcal{S}_0 \subset \mathcal{S}$ over $\mathcal{T}$, i.e., $\exists t_0 \in \mathcal{T}, \forall t \geq t_0, Y(t) \in \mathcal{S}_0$. Note that the boundary states can be generalized to any spatially marginalized state-related trajectories. More formally, we define *boundary feasibility* as follows in PDE dynamics.

Definition 1 (Boundary Feasibility for Finite-time Constraint Satisfaction) *With state $u(x,t)$ subjected to closed-loop PDE dynamics in Equation (1) with the boundary control input $U(t)$, the boundary control output $Y(t)$ is defined to be feasible over $\mathcal{T}$ within the given user-specified safe set $\mathcal{S}_0 \in \mathcal{S}$ if the following holds,*

$$\exists t_0 \in \mathcal{T}, \forall t_0 \leq t \leq T, Y(t) := u(\mathbf{0}, t) \in \mathcal{S}_0, \text{ where } u(\mathbf{1}, t) = U(t), u(x,0) \equiv U(0). \tag{2}$$

With boundary input and output trajectory pairs $\{[U_k(t), Y_k(t)], k = 1, 2, \dots, K\}$ from the unknown PDE dynamics,we formulate the problem for this paper as follows.

Problem 1 *Given K collected boundary input and output trajectory pairs $\{[U_{k,m}, Y_{k,m}], k = 1, 2, \dots, K, m = 1, 2, \dots, M\}$ with M-point temporal discretization, under consistent initial condition $u_k(x,0) \equiv U_k(0)$ from unknown but time-invariant PDE dynamics in Equation (1), we aim to find boundary control input $U(x)$ that guarantees boundary feasibility of boundary output $Y(t)$ with user-specified safe set $\mathcal{S}_0$ in Definition 1.*

3. Methodology

3.1. Neural Barrier Function for PDE Boundary Control

Boundary feasibility aims to find control input $U(t)$ for the constraint satisfaction of the marginalized output boundary $Y(t) := u(\mathbf{0}, t)$ from the underlying PDE dynamics with spatially-continuous unobservable state $u(x,t)$, which is challenging for conventional state-dependent-only CBFs. Hence, inspired by Garg and Panagou (2021b), we propose the neural boundary control barrier function (neural BCBF), explicitly incorporating time t into neural network parameterized function $\phi(t,Y) : \mathcal{T} \times \mathcal{S} \to \mathbb{R}$ for the time-dependent zero-sublevel set $\mathcal{S}_{\phi,t} := \{Y(t) \mid \phi(t,Y(t)) \le 0\}$. Note that the conventional CBF $\phi(Y)$ can be viewed as a specially case of BCBF $\phi(t,Y)$ where t remains constant. Another challenge is that the boundary feasibility in Equation (2) for PDE boundary control is defined on finite time domain $\mathcal{T} = [0,T]$, which requires a higher convergence rate to the safe set than the original asymptotic CBF (Ames et al., 2014) like fixed-time stability in Polyakov (2011); Garg and Panagou (2021a). The following theorem shows the feasibility of boundary control output $Y(t)$ within the user-specified safe set $\mathcal{S}_0$ under control signal $U(t)$.

Theorem 2 (Boundary Feasibility with Boundary Control Barrier Function) *For the state $u(x,t)$ from the closed-loop PDE dynamics with boundary control input $U(t) = u(\mathbf{1},t)$, $u(x,0) \equiv U_0$, the boundary feasibility of boundary output $Y(t) = u(\mathbf{0},t)$ over $\mathcal{T} = [0,T]$ within user-specified safe set $\mathcal{S}_0$ is guaranteed with neural BCBF $\phi(t,Y)$ if the following holds $\forall t \in \mathcal{T}$*

$$(\mathcal{S}_{\phi,t} := \{Y \mid \phi(t,Y) \le 0\} \subseteq \mathcal{S}_0) \bigwedge \left(\partial_Y \phi \cdot \frac{dY}{dt} + \partial_t \phi + \alpha\phi(t,Y) + C_{\alpha,T}\phi(0,U_0) \le 0\right), \quad (3)$$

where $C_{\alpha,T} := \frac{\alpha}{e^{\alpha T}-1} > 0$ is a constant for finite-time convergence.

Proof With the sublevel set $\mathcal{S}_{\phi,t}$ being the subset of $\mathcal{S}_0$, i.e., $\mathcal{S}_{\phi,t} := \{Y \mid \phi(t,Y) \le 0\} \subseteq \mathcal{S}_0$, it is sufficient to prove $\exists t_0 \in [0,T], s.t. \forall t \in [t_0,T], \phi(t,Y(t)) \le 0$. Now denote $\psi(t) := \phi(t,Y(t))$, by initial constant boundary condition $Y(0) = U_0$, the following equivalent inequalities hold,

$$\partial_Y \phi \cdot \frac{dY}{dt} + \partial_t \phi + \alpha\phi(t,Y) + C_{\alpha,T}\phi(0,Y(0)) \le 0 \iff \frac{d(e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t})}{dt} \le 0$$

So the function $e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t}$ is non-increasing over $t \in [0,T]$. By $T > 0$, we have

$$[e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t}]|_{t=T} < [e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t}]|_{t=0} \iff \psi(T) = \phi(T,Y(T)) < 0$$

So at least at $t_0 = T$, $\phi(t_0,Y(t_0)) < 0$, which concludes the proof of boundary feasibility of boundary output $Y(t) = u(\mathbf{0},t)$ over $\mathcal{T} = [0,T]$ in Definition 1. ■

The full proof can be found in Section B.2 in Hu and Liu (2025). Note that if $\phi(0,U_0) \le 0$, the forward invariance (Ames et al., 2019) can be obtained via $T \to \infty$. With the M-point temporal discretization of collected boundary input and output trajectory $\{[U_{k,m}, Y_{k,m}], k = 1,\ldots,K, m = 1,\ldots,M\}$, $\mathcal{S}_{\phi,t} \subseteq \mathcal{S}_0$ in Equation (3) induces the loss below following Dawson et al. (2022)

$$\mathcal{L}_{\mathcal{S}} = \sum_{k=1}^{K} \sum_{Y_{k,m} \in \mathcal{S}_0} [\phi(t_m, Y_{k,m})]_+ + \sum_{k=1}^{K} \sum_{Y_{k,m} \notin \mathcal{S}_0} [-\phi(t_m, Y_{k,m})]_+, \text{ with } [\cdot]_+ := \max\{0,\cdot\}. \quad (4)$$

However, it is challenging to find ${dY(t)}/{dt}$ involved in Equation (3) over the discrete time samples since the boundary output $Y(t) = u(\mathbf{0},t)$ is governed by the unknown closed-loop PDE dynamics with the boundary condition $U(t) = u(\mathbf{1},t)$. Besides, it is also non-trivial to find the boundary feasibility condition over boundary control input $U(t)$ for safety filtering due to non-Markov property. Therefore, we adopt the neural operator to learn the boundary input-output mapping as a neural transfer function to further mitigate the non-Markov issue in PDE boundary control problems with unknown PDE dynamics.

3.2. Learning Neural Operator for Input-output Boundary Mapping

Different from current applications of neural operators in learning PDE solutions by temporal mapping (Li et al., 2020a,b, 2022), we propose to adopt neural operator $\mathcal{G}_\theta : \{U : \mathcal{T} \to \mathcal{S}\} \mapsto \{Y : \mathcal{T} \to \mathcal{S}\}$ to model the spatial boundary mapping from input to output of the unknown closed-loop PDE dynamics in Equation (1), i.e., $Y(t) = u(\mathbf{1}, t) = \mathcal{G}_\theta(U)(t) = \mathcal{G}_\theta(u(\mathbf{0}, t))(t)$. Following Kovachki et al. (2023) under the setting of same Lebesgue-measurable domain $\mathcal{T}$ for hidden layers, the neural operator is defined as $\mathcal{G}_\theta = \mathcal{Q} \circ \mathcal{I}_{L-1} \circ \cdots \circ \mathcal{I}_0 \circ \mathcal{P}$, including pointwise lifting mapping $\mathcal{P} : \{U : \mathcal{T} \to \mathcal{S}\} \mapsto \{v_0 : \mathcal{T} \to \mathbb{R}^{d_{v_0}}\}$, iterative kernel integration layers $\mathcal{I}_l : \{v_l : \mathcal{T} \to \mathbb{R}^{d_{v_l}}\} \mapsto \{v_{l+1} : \mathcal{T} \to \mathbb{R}^{d_{v_{l+1}}}\}, l = 0, \ldots, L-1$, and the pointwise projection mapping $\mathcal{Q} : \{v_L : \mathcal{T} \to \mathbb{R}^{d_{v_L}}\} \mapsto \{Y : \mathcal{T} \to \mathcal{S}\}$. Specifically, the l-th kernel integration layer follows the following form with commonly-used integral kernel operator (Li et al., 2020a,b, 2022),

$$v_{l+1}(t) = \mathcal{I}_l(v_l)(t) = \sigma_{l+1}\left(W_l v_l(t) + \int_{\mathcal{T}} \kappa^{(l)}(t,s) v_l(s) ds + b_l(t)\right), l = 0, 1, \ldots, L-1, \quad (5)$$

where $\sigma_{l+1} : \mathbb{R}^{d_{v_{l+1}}} \to \mathbb{R}^{d_{v_{l+1}}}$ is the activation function, $W_l \in \mathbb{R}^{d_{v_{l+1}} \times d_{v_l}}$ is the local linear operator, $\kappa^{(l)} \in C(\mathcal{T} \times \mathcal{T}; \mathbb{R}^{d_{v_{l+1}} \times d_{v_l}})$ is the kernel function for integration, and $b_l \in C(\mathcal{T}; \mathbb{R}^{d_{v_{l+1}}})$ is the bias function. Besides, since lifting and projection operators $\mathcal{P}, \mathcal{Q}$ are pointwise local maps as special Nemitskiy operators (Dudley et al., 2011; Kovachki et al., 2023), i.e. there exist equivalent functions $P : \mathcal{S} \to \mathbb{R}^{d_{v_0}}, Q : \mathbb{R}^{d_{v_L}} \to \mathcal{S}$ such that $\mathcal{P}(U)(t) = P(U(t)), \mathcal{Q}(v_L)(t) = Q(v_L(t)), \forall t \in \mathcal{T}$. Therefore, combining Equation (5), we explicitly show the boundary mapping from control input $U(t)$ to output $Y(t)$ below, making them possible to be directly connected as $Y(t) = \mathcal{G}_\theta(U)(t)$,

$$Y(t) = \mathcal{G}_\theta(U)(t) = Q(v_L(t)), v_{l+1}(t) = \mathcal{I}_l(v_l)(t) \text{ in Equation (5)}, v_0(t) = P(U(t)), \quad (6)$$

where $P, Q, W_l, \kappa^{(l)}, b_l, l = 0, 1, \ldots, L-1$ parameterized with neural networks θ and compose the neural operator $Y(t) = G_\theta(U)(t)$. Given boundary input and output M-step temporally discretized K trajectory pairs $\{[U_{k,m}, Y_{k,m}], k = 1, 2, \ldots, K, m = 1, 2, \ldots, M\}$, G_θ and neural BCBF ϕ can be optimized together based on empirical-risk minimization using the following loss function,

$$\min_{\theta,\phi} \lambda_\mathcal{G}\mathcal{L}_\mathcal{G} + \lambda_\mathcal{S}\mathcal{L}_\mathcal{S} + \lambda_{BF}\mathcal{L}_{BF}, \text{ where } \mathcal{L}_\mathcal{G} = \sum_{k=1}^{K}\sum_{m=1}^{M} \|Y_{k,m} - \mathcal{G}_\theta(U_k)(t_m)\|^2, \mathcal{L}_\mathcal{S} \text{ in eq. (4)},$$

$$\mathcal{L}_{BF} = \sum_{k=1}^{K}\sum_{m=1}^{M} [\partial_{Y_{k,m}}\phi \cdot \frac{d\mathcal{G}_\theta(U_k)(t)}{dt} |_{t=t_m} + \partial_{t_m}\phi + \alpha\phi(t_m, Y_{k,m}) + C_{\alpha,T}\phi(0, U_{k,0})]_+, \quad (7)$$

and $[\cdot]_+ := \max\{0, \cdot\},, \lambda_\mathcal{G}, \lambda_\mathcal{S}, \lambda_{BF}$ are weight hyperparameters for $\mathcal{L}_\mathcal{G}, \mathcal{L}_\mathcal{S}, \mathcal{L}_{BF}$, respectively. The loss for neural operator learning $\mathcal{L}_\mathcal{G}$ is based on Equation (6), and the boundary feasibility (BF) loss of $\mathcal{L}_{BF}$ is based on Equation (3) with the replacement of ${dY(t)}/{dt}$ with ${d\mathcal{G}_\theta(U)(t)}/{dt}$, which will be detailed in the next section.

3.3. Safety Filtering with Quadratic Programming

Once the boundary input-output mapping is modeled by neural operator $\mathcal{G}_\theta$, the boundary output $Y(t)$ is directly related to boundary input $U(t)$ from trajectory to trajectory, bypassing the non-Markov property and the unknown closed-loop dynamics in Equation (1). We first find the derivative of boundary output $Y(t)$ w.r.t t based on neural operator $Y(t) = \mathcal{G}_\theta(U)(t)$. Applying chain rule to Equation (6), the following derivatives hold,

$$\frac{dY(t)}{dt} = \nabla Q^\top \frac{dv_L(t)}{dt}, \frac{dv_{l+1}(t)}{dt} = \mathcal{J}_l(\frac{dv_l}{dt})(t), l = L-1, \ldots, 0, \frac{v_0(t)}{dt} = \nabla P^\top \frac{dU(t)}{dt}, \tag{8}$$

where the derivative of kernel integration layer $\mathcal{J}_l : \{\frac{v_l}{dt} : \mathcal{T} \to \mathbb{R}^{d_{v_l}}\} \mapsto \{\frac{v_{l+1}}{dt} : \mathcal{T} \to \mathbb{R}^{d_{v_{l+1}}}\}, l = 0, 1, \ldots, L-1$ can be found through the derivative of Equation (5) in a recursive form below,

$$\frac{dv_{l+1}(t)}{dt} = \mathcal{J}_l(\frac{dv_l}{dt})(t) = \text{Diag}(\sigma'_{l+1}) \left(W_l \frac{dv_l(t)}{dt} + \int_{\mathcal{T}} \frac{\partial \kappa^{(l)}(t,s)}{\partial t} v_l(s) ds + \frac{db_l(t)}{dt} \right). \tag{9}$$

By combining Equation (8) and Equation (9), we have the following theorem to show how the boundary control input $U(t)$ can be chosen to guarantee the boundary feasibility of boundary output $Y(t)$ modeled by neural operator $\mathcal{G}_\theta$.

Theorem 3 (Boundary Feasibility with Neural Operator) *Assuming the neural operator $\mathcal{G}_\theta$ as an exact map from boundary input $U(t)$ to output $Y(t)$ for an unknown closed-loop PDE dynamics without model mismatch, the boundary control input $U(t)$ is guaranteed to induce boundary feasibility of output $Y(t)$ over $\mathcal{T} = [0, T]$ within the sublevel set of neural BCBF ϕ if $U(t)$ satisfies*

$$\partial_Y \phi(t, \mathcal{G}_\theta(U)) \frac{d\mathcal{G}_\theta(U)(t)}{dt} + \partial_t \phi(t, \mathcal{G}_\theta(U)) + \alpha\phi(t, \mathcal{G}_\theta(U)) + C_{\alpha,T}\phi(0, U(0)) \leq 0, \forall t \in \mathcal{T} \tag{10}$$

where $C_{\alpha,T} = \frac{\alpha}{e^{\alpha T}-1}$, and $\frac{d\mathcal{G}_\theta(U)(t)}{dt}$ can be found below with $\prod_1^0(\cdot) := 1$,

$$\frac{d\mathcal{G}_\theta(U)(t)}{dt} = \nabla Q^\top \prod_{l=0}^{L-1} \left(\mathit{Diag}(\sigma'_{L-l}) W_{L-1-l}\right) \nabla P^\top \frac{dU(t)}{dt} + \nabla Q^\top \mathit{Diag}(\sigma'_L) \sum_{i=0}^{L-1} \left([\prod_{j=1}^{i} W_{L-j} \mathit{Diag}(\sigma'_{L-j})] \left(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-1-i)}(t,s)}{\partial t} v_{L-1-i}(s) ds + \frac{db_{L-1-i}(t)}{dt} \right) \right) = \Lambda_\theta(t)\dot{U}(t) + \mu_\theta(t). \tag{11}$$

The proof of Equation (10) is based on Theorem 3 and Equation (11) can be derived by recursively applying Equation (9) to Equation (8). Please check Section B.3 of Hu and Liu (2025) for full proof.

Remark 4 *We remark that if the sublevel set of neural BCBF ϕ is a subset of user-specified safe set $\mathcal{S}_0$, and there is no model mismatch between neural operator $Y(t) = \mathcal{G}_\theta(U)(t)$ and unknown closed-loop PDE dynamics, Theorem 3 is equivalent to Theorem 2. Then the boundary control input $U(t)$ satisfying Equation (10) is guaranteed to induce the boundary feasibility of boundary output $Y(t)$ within the user-specified safe set $\mathcal{S}_0$.*

Based on the affine property of $\dot{U}(t)$ in Equation (11), we formulate the following quadratic programming with neural BCBF ϕ and neural operator $\mathcal{G}_\theta$ as a safety filter for $\dot{U}_{\text{nominal}}(t), \forall t \in \mathcal{T}$,

$$\dot{U}_{\text{safe}}(t) = \arg\min_{\dot{U} \in \mathbb{R}} \|\dot{U} - \dot{U}_{\text{nominal}}(t)\| \tag{12}$$

$$s.t.\ \partial_Y \phi(t, Y) \left(\Lambda_\theta(t)\dot{U} + \mu_\theta(t)\right) + \partial_t \phi(t, Y) + \alpha\phi(t, Y) + C_{\alpha,T}\phi(0, U_{\text{nominal}}(0)) \leq 0, \tag{13}$$

Algorithm 1 Safety Filtering Procedure for Discrete-time Implementation

```
1: Input: Initial and nominal control input $U^{\text{nominal}}_{0:M}$, neural operator $\mathcal{G}$, neural BCBF $\phi$
2: Output: Filtered safe control input $U^{\text{safe}}_{1:M}$
3: Initialize $\Delta U^{\text{safe}}_{1:M} = \Delta U^{\text{nominal}}_{1:M} \leftarrow U^{\text{nominal}}_{1:M} - U^{\text{nominal}}_{0:M-1}$, $Y^{\text{predict}}_{1:M} \leftarrow \mathcal{G}(U^{\text{nominal}}_{1:M})$
4: for $m = 1 : M$ do
5:     Find $\Delta U^{\text{safe}}_{m}$ through QP in Equation (12) based on $\Delta U^{\text{nominal}}_{m}$, $Y^{\text{predict}}_{1:M}$, $\mathcal{G}, \phi, U^{\text{nominal}}_{0}$
6:     Update $U^{\text{safe}}_{1:M} \leftarrow \sum_{i=1}^{m} \Delta U^{\text{safe}}_{i} + U^{\text{nominal}}_{0}$
7:     Update $Y^{\text{predict}}_{1:M} \leftarrow \mathcal{G}(U^{\text{safe}}_{1:M})$
8: end for
9: return $U^{\text{safe}}_{1:M}$
```

where $C_{\alpha,T} = \frac{\alpha}{e^{\alpha T}-1}$ and $\Lambda_\theta(t), \mu_\theta(t)$ can be found in Equation (11). Based on $\dot{U}_{\text{safe}}(t)$ at each step t, we update the potential boundary control input $U_{\text{safe}}(t)$ as $U_{\text{safe}}(t) = \int_0^t \dot{U}_{\text{safe}}(\tau)d\tau + U_{\text{nominal}}(0)$, so that the predicted boundary output $Y_{\text{predict}}(t) = \mathcal{G}_\theta(U_{\text{safe}})(t)$ can be found by the neural operator $\mathcal{G}_\theta$. Therefore, the next QP update can be solved for $\dot{U}_{\text{safe}}$ at the next time by Equation (12). Note that we let $\dot{U}_{\text{safe}} = \dot{U}_{\text{nominal}}$ for the unfiltered time steps during the QP iteration. The discrete-time implementation of the safety filtering procedure is shown in Algorithm 1. To accommodate the advection-dominated problems like the 1D hyperbolic problem or Navier Stokes, where the propagation speed from input to output boundary is not infinite, we predict the whole input and potentially delayed output trajectory through the neural operator at each step during the safety filtering. We adopt the predicted $Y(t)$ from the neural operator after each filtering step, and the filtering threshold is detailed as a workaround for the model mismatch in Section C of Hu and Liu (2025), along with a discussion on how approximation errors affect safety filtering.

4. Experiment

In this section, we aim to answer the following two questions: How does the proposed plug-and-play safety filtering perform based on the vanilla and constrained RL controllers in unknown PDE dynamics? How do different types of barrier functions, convergence criteria, and neural operator modeling influence the performance of the proposed safety filtering? We answer the first question in Section 4.2 and the second one in Section 4.3, following the setup of model training and evaluation metrics. Section C in Hu and Liu (2025) gives more details and results.

4.1. Experimental Setup

Environments and model-free controllers. We adopt the challenging PDE boundary control environments and the model-free reinforcement learning (RL) models from Bhan et al. (2024) to conduct our experiment. More specifically, the three environments include the unstable 1D hyperbolic (transport) equation, 1D parabolic (reaction-diffusion) equation and 2D nonlinear Navier-Stokes equation, where the last one is for tracking task and others are for stabilization task. Since our setting in Problem 1 does not have prior to the PDE equations, we choose the vanilla PPO (Schulman et al., 2017) and SAC (Haarnoja et al., 2018), and constrained RL models CPO (Achiam et al., 2017) and SAC-Lag (Ha et al., 2020) as the baselines in each environment for fair comparisons. The boundary control inputs are consistent with Bhan et al. (2024). For 1D environments, the boundary output for the hyperbolic PDE is $Y(t) = u(0,t)$ and the boundary output for the parabolic PDE

Table 1: Comparison of vanilla models w/o and w/ safety filtering under multiple environments.

1D hyperbolic equation	Reward (mean±std) (starting at ∼-300)	Feasible Rate (100 episodes)	Average Feasible Steps (50 control steps)
PPO in Bhan et al. (2024)	157.9±37.5	0.63	7.6
PPO with filtering	165.0±43.7	**0.71**	**9.8**
SAC in Bhan et al. (2024)	106.2±98.7	0.78	12.4
SAC with filtering	103.4±96.4	**0.85**	**13.9**
1D parabolic equation	Reward (mean±std) (starting at ∼0)	Feasible Rate (100 episodes)	Average Feasible Steps (1000 control steps)
PPO in Bhan et al. (2024)	164.5±20.7	0.60	155.0
PPO with filtering	168.2±23.5	**0.81**	**507.0**
SAC in Bhan et al. (2024)	156.5±6.2	0.72	118.4
SAC with filtering	157.5±6.8	**0.87**	**449.8**
2D Navier-Stokes equation	Reward (mean±std) (starting at ∼-100)	Feasible Rate (100 episodes)	Average Feasible Steps (200 control steps)
PPO in Bhan et al. (2024)	-5.37±0.01	0.86	2.0
PPO with filtering	-5.72±0.17	**0.99**	**32.0**
SAC in Bhan et al. (2024)	-18.05±1.13	0.80	17.5
SAC with filtering	-18.36±1.25	**0.85**	**21.3**

Table 2: Comparison of constrained RL models w/o and w/ safety filtering for 1D hyperbolic PDE.

Constrained RL Models	Reward (mean±std) (starting at ∼-300)	Feasible Rate under Y constraints (100 episodes)		Average Feasible Steps (50 control steps)	
CPO (Achiam et al., 2017)	168.7±28.8	0.88 (Y<1)	0.52(Y<0)	11.2 (Y<1)	4.2 (Y<0)
CPO with filtering	168.8±28.6	**0.89** (Y<1)	**0.56** (Y<0)	**14.8** (Y<1)	**4.7** (Y<0)
SAC-Lag (Ha et al., 2020)	110.9±92.1	0.84 (Y<0)	0.50 (Y<-0.5)	**20.8** (Y<0)	**3.1** (Y<-0.5)
SAC-Lag with filtering	107.6±90.3	**0.90** (Y<0)	**0.67** (Y<-0.5)	18.9 (Y<0)	2.9 (Y<-0.5)

$Y(t) = u(0.5, t)$. For the 2D environment, the boundary output is $Y(t) = u(0.5, 0.95, t)$, which has the maximum speed over 2D plane. The boundary feasibility constraints are detailed in Section 3.1 of Hu and Liu (2025). With the PDE controllers in Bhan et al. (2024), we collect 50k pairs of boundary input $U(t)$ and output $Y(t)$ trajectory with safety labels based on safety constraints. The resolution of collected trajectories is consistent with the control frequency of each environment.

Model training and evaluation metrics. With the collected dataset from vanilla RL models, we adopt the Fourier neural operator (FNO) (Li et al., 2020a) as the default neural operator model and train it with Markov neural operator (MNO) (Li et al., 2022) using the default hyper-parameters. For the neural BCBF training, following Zhang et al. (2023); Hu et al. (2024), we use a 4-layer feedforward neural network with ReLU activations to parameterize BCBFs and incorporate Equation (4) and Equation (7) with default $\alpha = 10^{-5}$ into the regular model training pipeline (Zhao et al., 2020; Dawson et al., 2022) to train time-dependent BCBF $\phi(t, Y)$ as default. With the well-trained neural operator and neural BCBF, we solve the QP of Equation (12) though CPLEX (IBM) For the evaluation of safety filtering for RL controllers, we keep the original RL rewards from Bhan et al. (2024) as a metric to show if the performance is compromised by the enhancement of safety constraints. Besides, we introduce two new metrics regarding boundary feasibility, *Feasible Rate* and *Average Feasible Steps*. *Feasible Rate* is the ratio of trajectories that boundary feasibility in Definition 1 is achieved, i.e., the boundary output falls into the safe set and will not go out of it by the end of a single trajectory with finite steps. *Average Feasible Steps* is the mean steps among boundary feasi-

Table 3: Comparison of time-independent and time-dependent safety filtering in hyperbolic equations.

Different safety filtering	Reward (mean±std) (starting at ∼-300)	Feasible Rate (100 episodes)	Average Feasible Steps (50 control steps)
PPO with filtering of $\phi(Y)$	162.3±44.5	0.63	8.3
PPO with filtering of $\phi(t, Y)$	165.0±43.7	**0.71**	**9.8**
SAC with filtering of $\phi(Y)$	103.3±98.4	0.57	**15.7**
SAC with filtering of $\phi(t, Y)$	103.4±96.4	**0.85**	13.9

Table 4: Filtering with BCBF $\phi(t, Y)$ under different neural operators for 1D hyperbolic equation.

Different neural operators	Reward (mean±std) (starting at ∼-300)	Feasible Rate (100 episodes)	Average Feasible Steps (50 control steps)
PPO w. MNO (Li et al., 2022)	163.8±47.2	**0.78**	9.0
PPO w. FNO (Li et al., 2020a)	165.0±43.7	0.71	**9.8**
SAC w. MNO (Li et al., 2022)	103.3±96.4	0.84	**14.7**
SAC w. FNO (Li et al., 2020a)	103.4±96.4	**0.85**	13.9

ble trajectories in which the boundary output is consistently kept in the safe set until the end of the trajectory, characterizing how long the boundary feasibility is achieved and maintained.

4.2. Results Comparison

Comparison of vanilla models with safety filtering. From all three PDE environments in Table 1, vanilla PPO and SAC with safety filtering outperform vanilla PPO and SAC in feasible rate and average feasible steps, demonstrating the effectiveness of safety filtering for boundary constraint satisfiability. Besides, the rewards in parabolic and hyperbolic equations can also be improved through filtering due to the alignment of boundary constraints and the stabilization goal. The reward of the filtered SAC model in the hyperbolic equation is compromised because the constraint $Y < 0$ conflicts with the stabilization task of $Y \to 0$. In the 2D Navier-Stokes PDE, due to the inconsistency between the specific high-speed point boundary for constraint and the full 2D plane for reward, boundary feasibility is enhanced by safety filtering while rewards are compromised.

Safety filtering performance based on constrained RL models. To further show the plug-and-play efficacy of our safety filtering method, we present the filtering performance over the constrained RL models in Table 2 using the pre-trained BCBF, which is trained over data collected from vanilla RL models. We can see that compared to CPO (Achiam et al., 2017), the filtered controller tends to improve the boundary feasibility, especially for the stronger constraint $Y < 0$. Safety filtering over SAC-Lag (Ha et al., 2020) will give higher feasibility rates over the boundary, while the average feasible steps slightly decrease because feasible steps along trajectories become more concentrated and less divergent after filtering. Besides, despite the potential conflict between boundary constraint and stabilization, the reward will not be hurt significantly via safety filtering.

4.3. Ablation Study

Comparison of safety filtering using $\phi(Y)$ vs. $\phi(t, Y)$. With different boundary control barrier functions in Table 3, with the PPO model, safety filtering with $\phi(t, Y)$ outperforms filtering with $\phi(Y)$ in reward and boundary feasibility metrics, showing that time-dependent BCBF can distinguish the feasibility of the PDE boundary state more effectively by explicitly taking time as an input compared to the time-independent one. Based on the vanilla SAC model, reward and feasible

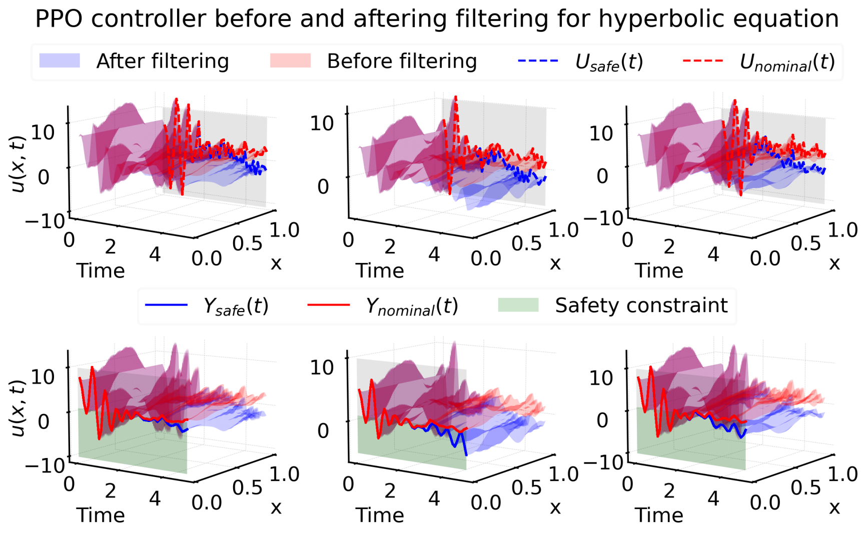

Figure 2: Visualization of three state trajectories $u(x,t)$ (left, mid, right) for hyperbolic equation under PPO controller with and without safety filtering. Boundary control inputs $U(t)$ are in dashed lines, and boundary outputs $Y(t)$ are in solid lines. The boundary constraint $Y(t) < 1$ is in green.

rate with $\phi(t, Y)$ filtering is higher but the average feasible step is lower than $\phi(Y)$ filtering, because time-independent BCBF $\phi(Y)$ tends to have divergent performance with more non-feasible trajectories and more feasible steps for feasible trajectories.

Boundary mapping with different neural operators. Here we compare two neural operators, FNO (Li et al., 2020a) and MNO (Li et al., 2022), for learning the boundary mapping from control input $U(t)$ to output $Y(t)$ for 1D hyperbolic equation in Table 4. With the same time-dependent BCBF $\phi(t, Y)$, the safety filtering with FNO presents higher rewards under both PPO and SAC base models, showing that FNO is more suitable for learning low-resolution trajectories with 50 sampled points. Besides, MNO shows a better feasible rate and average feasible steps performance, especially with SAC as the base model, since the MNO model has a larger model complexity.

Qualitative visualization. In this section, we visualize and compare multiple trajectories under the 1D hyperbolic equation using the PPO controller without and with safety filtering of $\phi(t, Y)$, as shown in Figure 2. We can see that for each trajectory, the state value $u(x,t)$ after filtering is lower than that before filtering. More specifically, as time goes by, the filtered control input $U(t)_{\text{safe}}$ in blue dashed lines deviates more away from nominal control input $U(t)_{\text{nominal}}$ in red dashed lines, causing the filtered boundary output $Y(t)_{\text{safe}}$ in blue solid lines to satisfy the constraint $Y(t) < 1$ compared to the nominal boundary output $Y(t)_{\text{nominal}}$ in red solid lines.

5. Conclusion

In this work, we introduce a novel safe PDE boundary control framework using safety filtering with neural certification. A neural operator and a boundary control barrier function are learned from collected PDE boundary input and output trajectories within a given safe set. We show that the change in the BCBF depends linearly on the change in the input boundary. Hence, safety filtering can be done by solving a quadratic programming problem to ensure the boundary feasibility. Experiments on three challenging PDE control environments validate the effectiveness of the proposed method in terms of both general performance and constraint satisfaction.

Acknowledgments

This work is partially supported by the National Science Foundation, Grant No. 2144489.

References

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE conference on decision and control*, pages 6271–6278. IEEE, 2014.

Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pages 3420–3431. IEEE, 2019.

Luke Bhan, Yuanyuan Shi, and Miroslav Krstic. Neural operators for bypassing gain and control computations in pde backstepping. *IEEE Transactions on Automatic Control*, 2023.

Luke Bhan, Yuexin Bian, Miroslav Krstic, and Yuanyuan Shi. Pde control gym: A benchmark for data-driven boundary control of partial differential equations. In *6th Annual Learning for Dynamics & Control Conference*. PMLR, 2024.

Nicholas Boffi, Stephen Tu, Nikolai Matni, Jean-Jacques Slotine, and Vikas Sindhwani. Learning stability certificates from data. In *Conference on Robot Learning*, pages 1341–1350. PMLR, 2021.

Steven L Brunton and J Nathan Kutz. Promising directions of machine learning for partial differential equations. *Nature Computational Science*, 4(7):483–494, 2024.

Ya-Chien Chang, Nima Roohi, and Sicun Gao. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

Charles Dawson, Zengyi Qin, Sicun Gao, and Chuchu Fan. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pages 1724–1735. PMLR, 2022.

Charles Dawson, Sicun Gao, and Chuchu Fan. Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control. *IEEE Transactions on Robotics*, 2023.

Richard M Dudley, Rimas Norvaiša, and Rimas Norvaiša. *Concrete functional calculus*. Springer, 2011.

Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Society, 1998.

Kunal Garg and Dimitra Panagou. Characterization of domain of fixed-time stability under control input constraints. In *2021 American Control Conference (ACC)*, pages 2272–2277. IEEE, 2021a.

Kunal Garg and Dimitra Panagou. Robust control barrier and control lyapunov functions with fixed-time convergence guarantees. In *2021 American Control Conference (ACC)*, pages 2292–2297. IEEE, 2021b.

Sehoon Ha, Peng Xu, Zhenyu Tan, Sergey Levine, and Jie Tan. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Eric Heiden, David Millard, Erwin Coumans, Yizhou Sheng, and Gaurav S Sukhatme. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9474–9481. IEEE, 2021.

Hanjiang Hu and Changliu Liu. Safe pde boundary control with neural operators. *arXiv preprint arXiv:2411.15643*, 2025. URL http://arxiv.org/abs/2411.15643.

Hanjiang Hu, Yujie Yang, Tianhao Wei, and Changliu Liu. Verification of neural control barrier functions with symbolic derivative bounds propagation. In *8th Annual Conference on Robot Learning*, 2024.

IBM. Ibm ilog cplex optimization studio. URL https://www.ibm.com/products/ilog-cplex-optimization-studio.

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Shumon Koga and Miroslav Krstic. Safe pde backstepping qp control with high relative degree cbfs: Stefan model with actuator dynamics. *IEEE Transactions on Automatic Control*, 68(12): 7195–7208, 2023.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Miroslav Krstic and Matt Bement. Nonovershooting control of strict-feedback nonlinear systems. *IEEE Transactions on Automatic Control*, 51(12):1938–1943, 2006.

Miroslav Krstic and Andrey Smyshlyaev. *Boundary control of PDEs: A course on backstepping designs*. SIAM, 2008.

Wuquan Li and Miroslav Krstic. Mean-nonovershooting control of stochastic nonlinear systems. *IEEE Transactions on Automatic Control*, 66(12):5756–5771, 2020.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Zongyi Li, Miguel Liu-Schiaffini, Nikola Kovachki, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Learning dissipative dynamics in chaotic systems. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 16768–16781, 2022.

Rayan Mazouz, Karan Muvvala, Akash Ratheesh Babu, Luca Laurenti, and Morteza Lahijanian. Safety guarantees for neural network dynamic systems via stochastic barrier functions. *Advances in Neural Information Processing Systems*, 35:9672–9686, 2022.

Lawrence Perko. *Differential equations and dynamical systems*, volume 7. Springer Science & Business Media, 1996.

Andrey Polyakov. Nonlinear feedback design for fixed-time stabilization of linear control systems. *IEEE transactions on Automatic Control*, 57(8):2106–2110, 2011.

Lorenz Pyta, Michael Herty, and Dirk Abel. Optimal feedback control of the incompressible navier-stokes-equations using reduced order models. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2519–2524. IEEE, 2015.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Esteban Samaniego, Cosmin Anitescu, Somdatta Goswami, Vien Minh Nguyen-Thanh, Hongwei Guo, Khader Hamdia, Xiaoying Zhuang, and Timon Rabczuk. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362:112790, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ji Wang and Miroslav Krstic. Safe adaptive control of hyperbolic pde-ode cascades. *arXiv preprint arXiv:2309.05596*, 2023.

Rose Yu and Rui Wang. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proceedings of the National Academy of Sciences*, 121(27):e2311808121, 2024.

Hongchao Zhang, Wu Junlin, Vorobeychik Yevgeniy, and Andrew Clark. Exact verification of reLU neural control barrier functions. In *Advances in neural information processing systems*, 2023.

Xiangyuan Zhang, Weichao Mao, Saviz Mowlavi, Mouhacine Benosman, and Tamer Başar. Controlgym: Large-scale control environments for benchmarking reinforcement learning algorithms. In *6th Annual Learning for Dynamics & Control Conference*, pages 181–196. PMLR, 2024a.

Xiangyuan Zhang, Saviz Mowlavi, Mouhacine Benosman, and Tamer Başar. Policy optimization for pde control with a warm start. *arXiv preprint arXiv:2403.01005*, 2024b.

Hengjun Zhao, Xia Zeng, Taolue Chen, and Zhiming Liu. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, pages 1–11, 2020.