Safe PDE Boundary Control with Neural Operators

Hanjiang Hu [1] **Changliu Liu** [1]

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.

[1]Robotics Institute, Carnegie Mellon University. Correspondence to: Hanjiang Hu <hanjianghu@cmu.edu>, Changliu Liu <cliu6@andrew.cmu.edu>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

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 & 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 & 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 & Smyshlyaev, 2008b). 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 & Bement, 2006; Li & Krstic, 2020; Koga & Krstic, 2023; Wang & 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

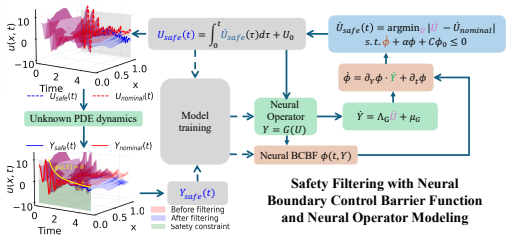

Figure 1: Overview of our safety filtering method for PDE boundary control with neural BCBF. Solid line arrows denote the safety filtering, while dashed ones denote the model training.

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 Appendix A. 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 $u \in \mathcal{S}$ 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,

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

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 2.1 (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,

$$\exists t_0 \in \mathcal{T}, \forall t_0 \leq t \leq T, Y(t) := u(\mathbf{0}, t) \in \mathcal{S}_0, \quad (2)$$

where $u(\mathbf{1}, t) = U(t), u(x, 0) \equiv U(0)$.

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 2.1.

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 & 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 $\mathcal{S}_{\phi,t} := \{Y(t) \mid \phi(t, Y(t)) \leq 0\}$. 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 & 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 3.1 (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 $\forall t \in \mathcal{T}$*

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

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

The full proof can be found in Appendix B.2. Note that if $\phi(0, U_0) \leq 0$, 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)

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

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),

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

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 $Y(t) = \mathcal{G}_\theta(U)(t)$,

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

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, $\min_{\theta,\phi} \lambda_{\mathcal{G}}\mathcal{L}_{\mathcal{G}} + \lambda_{\mathcal{S}}\mathcal{L}_{\mathcal{S}} + \lambda_{BF}\mathcal{L}_{BF}$, where

$$\mathcal{L}_{\mathcal{G}} = \sum_{k=1}^{K}\sum_{m=1}^{M} \|Y_{k,m} - \mathcal{G}_\theta(U_k)(t_m)\|^2, \mathcal{L}_{\mathcal{S}} \text{ in Equation (4)},$$

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

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,

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

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,

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

By combining Equation (8) and Equation (9), we have the theorem in Appendix B.3 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 $\mathcal{G}_\theta$. With the affine property of $\dot{U}(t)$ in $\frac{d\mathcal{G}_\theta(U)(t)}{dt} = \Lambda_\theta(t)\dot{U} + \mu_\theta(t)$, we formulate the following quadratic programming with neural BCBF ϕ and neural operator $\mathcal{G}_\theta$ as a safety filter for $\dot{U}_{\text{nominal}}(t), \forall t \in \mathcal{T}$,

$$\dot{U}_{\text{safe}}(t) = \arg\min_{\dot{U}\in\mathbb{R}} \|\dot{U} - \dot{U}_{\text{nominal}}(t)\| \quad (10)$$

$$s.t.\ \partial_Y\phi(t,Y)\left(\Lambda_\theta(t)\dot{U} + \mu_\theta(t)\right) + \partial_t\phi(t,Y) + \alpha\phi(t,Y) + C_{\alpha,T}\phi(0, U_{\text{nominal}}(0)) \le 0, \quad (11)$$

where $C_{\alpha,T} = \frac{\alpha}{e^{\alpha T}-1}$ and $\Lambda_\theta(t), \mu_\theta(t)$ can be found in Equation (31). 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 (10). More details of discrete-time implementation can be found in Appendix D.1.

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**

4. Experiment

Following (Bhan et al., 2024), we conduct the safety filtering on the RL controllers of PPO and SAC on hyperbolic, parabolic and Navier-Stokes PDEs. 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. 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. The detailed full version can be found in Appendices C and D.

5. Conclusion

We introduce a novel safe PDE boundary control framework using safety filtering based on neural operator modeling. Experiments on three challenging PDE control environments validate the effectiveness of the proposed method.

References

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. PMLR, 2017.

Agrawal, D. R. and Panagou, D. Safe control synthesis via input constrained control barrier functions. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 6113–6118. IEEE, 2021.

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

Ames, A. D., Coogan, S., Egerstedt, M., Notomista, G., Sreenath, K., and Tabuada, P. Control barrier functions: Theory and applications. In *2019 18th European control conference (ECC)*, pp. 3420–3431. IEEE, 2019.

Bansal, S. and Tomlin, C. J. Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1817–1824. IEEE, 2021.

Bhan, L., Shi, Y., and Krstic, M. Neural operators for bypassing gain and control computations in pde backstepping. *IEEE Transactions on Automatic Control*, 2023.

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

Boffi, N., Tu, S., Matni, N., Slotine, J.-J., and Sindhwani, V. Learning stability certificates from data. In *Conference on Robot Learning*, pp. 1341–1350. PMLR, 2021.

Botteghi, N. and Fasel, U. Parametric pde control with deep reinforcement learning and differentiable l0-sparse polynomial policies. *arXiv preprint arXiv:2403.15267*, 2024.

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

Chang, Y.-C., Roohi, N., and Gao, S. Neural lyapunov control. *Advances in neural information processing systems*, 32, 2019.

Cheng, H., Hu, H., and Liu, C. Robust tracking control with neural network dynamic models under input perturbations. *arXiv preprint arXiv:2410.10387*, 2024.

Choi, J. J., Lee, D., Sreenath, K., Tomlin, C. J., and Herbert, S. L. Robust control barrier–value functions for safety-critical control. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pp. 6814–6821. IEEE, 2021.

Dai, B., Krishnamurthy, P., and Khorrami, F. Learning a better control barrier function. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 945–950. IEEE, 2022.

Dawson, C., Qin, Z., Gao, S., and Fan, C. Safe nonlinear control using robust neural lyapunov-barrier functions. In *Conference on Robot Learning*, pp. 1724–1735. PMLR, 2022.

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

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

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

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

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

Ha, S., Xu, P., Tan, Z., Levine, S., and Tan, J. Learning to walk in the real world with minimal human effort. *arXiv preprint arXiv:2002.08550*, 2020.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

Heiden, E., Millard, D., Coumans, E., Sheng, Y., and Sukhatme, G. S. Neuralsim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9474–9481. IEEE, 2021.

Hsu, K.-C., Hu, H., and Fisac, J. F. The safety filter: A unified view of safety-critical control in autonomous systems. *Annual Review of Control, Robotics, and Autonomous Systems*, 7, 2023.

Hu, H., Yang, Y., Wei, T., and Liu, C. Verification of neural control barrier functions with symbolic derivative bounds propagation. In *8th Annual Conference on Robot Learning*, 2024.

Hu, P., Qian, X., Deng, W., Wang, R., Feng, H., Feng, R., Zhang, T., Wei, L., Wang, Y., Ma, Z.-M., et al. From uncertain to safe: Conformal fine-tuning of diffusion models for safe pde control. *arXiv preprint arXiv:2502.02205*, 2025.

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

Kochkov, D., Smith, J. A., Alieva, A., Wang, Q., Brenner, M. P., and Hoyer, S. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Koga, S. and Krstic, M. 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.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

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

Krstic, M. and Smyshlyaev, A. Backstepping boundary control for first-order hyperbolic pdes and application to systems with actuator and sensor delays. *Systems & Control Letters*, 57(9):750–758, 2008a.

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

Krstic, M., Bhan, L., and Shi, Y. Neural operators of backstepping controller and observer gain functions for reaction–diffusion pdes. *Automatica*, 164:111649, 2024.

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

Li, Y., Sun, Y., Ma, P., Sifakis, E., Du, T., Zhu, B., and Matusik, W. Neuralfluid: Neural fluidic system design and control with differentiable simulation. *arXiv preprint arXiv:2405.14903*, 2024a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020b.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020c.

Li, Z., Liu-Schiaffini, M., Kovachki, N., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Learning dissipative dynamics in chaotic systems. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 16768–16781, 2022.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024b.

Lindemann, L. and Dimarogonas, D. V. Control barrier functions for signal temporal logic tasks. *IEEE control systems letters*, 3(1):96–101, 2018.

Lindemann, L., Hu, H., Robey, A., Zhang, H., Dimarogonas, D., Tu, S., and Matni, N. Learning hybrid control barrier functions from data. In *Conference on Robot Learning*, pp. 1351–1370. PMLR, 2021.

Liu, C. and Tomizuka, M. Control in a safe set: Addressing safety in human-robot interactions. In *Dynamic Systems and Control Conference*, volume 46209, pp. V003T42A003. American Society of Mechanical Engineers, 2014.

Liu, S., Liu, C., and Dolan, J. Safe control under input limits with neural control barrier functions. In *Conference on Robot Learning*, pp. 1970–1980. PMLR, 2022a.

Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, S., Li, B., and Zhao, D. Constrained variational policy optimization for safe reinforcement learning. In *International Conference on Machine Learning*, pp. 13644–13668. PMLR, 2022b.

Liu, Z., Guo, Z., Cen, Z., Zhang, H., Yao, Y., Hu, H., and Zhao, D. Towards robust and safe reinforcement learning with benign off-policy data. In *International Conference on Machine Learning*, pp. 21586–21610. PMLR, 2023.

Liu, Z., Guo, Z., Lin, H., Yao, Y., Zhu, J., Cen, Z., Hu, H., Yu, W., Zhang, T., Tan, J., et al. Datasets and benchmarks for offline safe reinforcement learning. *Journal of Data-centric Machine Learning Research*, 2024.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. E. Learning nonlinear operators via deeponet based on the

universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.

Ma, P., Chen, P. Y., Deng, B., Tenenbaum, J. B., Du, T., Gan, C., and Matusik, W. Learning neural constitutive laws from motion observations for generalizable pde dynamics. In *International Conference on Machine Learning*, pp. 23279–23300. PMLR, 2023.

Manda, L., Chen, S., and Fazlyab, M. Learning performance-oriented control barrier functions under complex safety constraints and limited actuation. *arXiv preprint arXiv:2401.05629*, 2024a.

Manda, L., Chen, S., and Fazlyab, M. Domain adaptive safety filters via deep operator learning. *arXiv preprint arXiv:2410.14528*, 2024b.

Mazouz, R., Muvvala, K., Ratheesh Babu, A., Laurenti, L., and Lahijanian, M. Safety guarantees for neural network dynamic systems via stochastic barrier functions. *Advances in Neural Information Processing Systems*, 35: 9672–9686, 2022.

Mowlavi, S. and Nabi, S. Optimal control of pdes using physics-informed neural networks. *Journal of Computational Physics*, 473:111731, 2023.

NeuralOperators.jl. Neuraloperators. URL https://github.com/SciML/NeuralOperators.jl.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

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

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

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

Qi, J., Zhang, J., and Krstic, M. Neural operators for delay-compensating control of hyperbolic pides. *Available at SSRN 4543896*, 2023.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., Zhuang, X., and Rabczuk, T. 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.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Seidman, J., Kissas, G., Perdikaris, P., and Pappas, G. J. Nomad: Nonlinear manifold decoders for operator learning. *Advances in Neural Information Processing Systems*, 35: 5601–5613, 2022.

Smyshlyaev, A. and Krstic, M. Closed-form boundary state feedbacks for a class of 1-d partial integro-differential equations. *IEEE Transactions on Automatic control*, 49 (12):2185–2202, 2004.

Smyshlyaev, A. and Krstic, M. *Adaptive control of parabolic PDEs*. Princeton University Press, 2010.

So, O., Serlin, Z., Mann, M., Gonzales, J., Rutledge, K., Roy, N., and Fan, C. How to train your neural control barrier function: Learning safety filters for complex input-constrained systems. *arXiv preprint arXiv:2310.15478*, 2023.

Soroco, M., Song, J., Xia, M., Emond, K., Sun, W., and Chen, W. Pde-controller: Llms for autoformalization and reasoning of pdes. *arXiv preprint arXiv:2502.00963*, 2025.

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

Wei, L., Hu, P., Feng, R., Feng, H., Du, Y., Zhang, T., Wang, R., Wang, Y., Ma, Z.-M., and Wu, T. A generative approach to control complex physical systems. *arXiv e-prints*, pp. arXiv–2407, 2024a.

Wei, T., Kang, S., Zhao, W., and Liu, C. Persistently feasible robust safe control by safety index synthesis and convex semi-infinite programming. *IEEE Control Systems Letters*, 7:1213–1218, 2022.

Wei, T., Marzari, L., Yun, K. S., Hu, H., Niu, P., Luo, X., and Liu, C. Modelverification. jl: a comprehensive toolbox for formally verifying deep neural networks. *arXiv preprint arXiv:2407.01639*, 2024b.

Xiao, W., Belta, C. A., and Cassandras, C. G. High order control lyapunov-barrier functions for temporal logic specifications. In *2021 American Control Conference (ACC)*, pp. 4886–4891. IEEE, 2021.

Xiao, W., Belta, C. A., and Cassandras, C. G. Sufficient conditions for feasibility of optimal control problems using control barrier functions. *Automatica*, 135:109960, 2022.

Xiao, W., Wang, T.-H., Hasani, R., Chahine, M., Amini, A., Li, X., and Rus, D. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 2023.

Yang, Y., Hu, H., Wei, T., Li, S. E., and Liu, C. Scalable synthesis of formally verified neural value function for hamilton-jacobi reachability analysis. *arXiv preprint arXiv:2407.20532*, 2024.

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

Zhang, H., Junlin, W., Yevgeniy, V., and Clark, A. Exact verification of reLU neural control barrier functions. In *Advances in neural information processing systems*, 2023.

Zhang, X., Mao, W., Mowlavi, S., Benosman, M., and Başar, T. Controlgym: Large-scale control environments for benchmarking reinforcement learning algorithms. In *6th Annual Learning for Dynamics & Control Conference*, pp. 181–196. PMLR, 2024a.

Zhang, X., Mowlavi, S., Benosman, M., and Başar, T. Policy optimization for pde control with a warm start. *arXiv preprint arXiv:2403.01005*, 2024b.

Zhao, H., Zeng, X., Chen, T., and Liu, Z. Synthesizing barrier certificates using neural networks. In *Proceedings of the 23rd international conference on hybrid systems: Computation and control*, pp. 1–11, 2020.

Zinage, V., Chandra, R., and Bakolas, E. Neural differentiable integral control barrier functions for unknown nonlinear systems with input constraints. *arXiv preprint arXiv:2312.07345*, 2023.

A. Related Work

Control for PDE Dynamics. PDE control problems can be in-domain control (Botteghi & Fasel, 2024; Zhang et al., 2024b) or boundary control (Krstic & Smyshlyaev, 2008b; Smyshlyaev & Krstic, 2010), where the latter is more commonly-seen setting in the real world. As it has been studied for over a decade, backstepping has become a dominant approach for boundary control with known PDE dynamics (Krstic & Smyshlyaev, 2008a; Smyshlyaev & Krstic, 2004). Recently, learning-based controllers have gotten rid of the requirement of analytical forms of unstable PDE dynamics and become a promising solution to the PDE control problems (Botteghi & Fasel, 2024; Zhang et al., 2024b; Krstic et al., 2024; Qi et al., 2023; Mowlavi & Nabi, 2023; Wei et al., 2024a; Soroco et al., 2025). Regarding the safety of constraint satisfaction in the PDE dynamics, current backstepping-based safe PDE control methods (Krstic & Bement, 2006; Li & Krstic, 2020; Koga & Krstic, 2023; Wang & Krstic, 2023) still assume the non-stable PDE dynamics is known. Recently, (Hu et al., 2025) introduce safe diffusion models for PDE Control based on conformal prediction to quantify uncertainty. Instead, we focus on boundary safety constraint satisfiability in the PDE boundary control signal without any prior knowledge of PDE dynamics.

Safe Control with Neural Certificate For the control of the ODE dynamical system, there is rich literature regarding learning-based controllers with safety guarantees or certificates (Boffi et al., 2021; Dawson et al., 2023; Xiao et al., 2023; Lindemann et al., 2021; Chang et al., 2019; Mazouz et al., 2022). Neural networks have been used to parameterize the CBFs under complex dynamics with bounded control inputs (Liu et al., 2022a; So et al., 2023; Zinage et al., 2023; Dawson et al., 2022; Dai et al., 2022), which result in forward invariance of the user-specified safe set to guarantee the safety with neural certificate for learning-based controllers (Choi et al., 2021; Wei et al., 2022; Agrawal & Panagou, 2021; Xiao et al., 2022; Hsu et al., 2023), i.e. once the states enter the safe set, they will never go out. However, forward invariance may not hold in the PDE boundary control setting with commonly-seen highly oscillating trajectories. For example, highly-oscillating trajectories may go out of the safe set during the early oscillation and break the forward invariance defined by conventional ODE CBFs (Liu & Tomizuka, 2014; Ames et al., 2014), but they could still converge to the constraint satisfaction by the end of time. Therefore, we focus on boundary feasibility, a new notion introduced in this paper. Approach-wise, the CBF-QP for ODE dynamics (Liu & Tomizuka, 2014; Lindemann & Dimarogonas, 2018; Xiao et al., 2021; Garg & Panagou, 2021b) does not apply. That is because PDE boundary control does not have Markov property at each control step, due to the infinite-dimensional unobserved non-boundary states. We adopt a neural operator to model the trajectory-to-trajectory mapping and control the change of input boundary through a novel QP formulation.

Neural Operator Learning for PDEs. Neural operator learning has become a powerful tool for solving PDEs by learning mappings between function spaces rather than pointwise approximations (Kovachki et al., 2023; Brunton & Kutz, 2024). Recent research has demonstrated the utility of neural operators in multiple science and engineering fields like fluid dynamics, weather forecasting, and robotics (Kochkov et al., 2021; Pathak et al., 2022; Heiden et al., 2021; Raissi et al., 2019). There exist multiple architectures for neural operators based on different mathematical properties of data. (Lu et al., 2021) introduces DeepONet with a branch and a trunk network, and NOMAD (Seidman et al., 2022) adopts nonlinear decoder map to learn submanifolds in function spaces, while Green's function-inspired neural operators (Li et al., 2020a;b;c; 2022; 2024b) adopt linear integral kernel representation with various kernel implementations. Learning-based methods (Ma et al., 2023; Li et al., 2024a) are proposed for differentiable simulation of PDE dynamics, but neural control of PDE dynamics is less explored. Recent work (Manda et al., 2024b) introduces operator learning for mapping from environmental parameters to the corresponding CBF under HJ-PDE (Bansal & Tomlin, 2021; Manda et al., 2024a), which does not directly study the PDE control problem. For the PDE boundary control problem, current works (Bhan et al., 2023; Krstic et al., 2024) only adopt neural operators to learn the integral kernel in backstepping, which does not release the full potential of neural operator for characterizing and controlling unknown dynamics. The proposed work is the first to leverage neural operators to learn the direct mapping from control input to boundary output as a transfer function.

B. Proofs

B.1. Preliminary

Definition B.1 (Boundary Feasibility for Finite-time Constraint Satisfaction)**.** (restated from Definition 2.1) 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,

$$\exists t_0 \in \mathcal{T}, \forall t_0 \leq t \leq T, Y(t) := u(\mathbf{0},t) \in \mathcal{S}_0, \text{ where } u(\mathbf{1},t) = U(t), u(x,0) \equiv U(0). \tag{12}$$

Definition B.2 (Neural operator for input-output boundary mapping)**.** Neural operator from Section 3.2 $\mathcal{G}_\theta : \{U : \mathcal{T} \to \mathcal{S}\} \mapsto \{Y : \mathcal{T} \to \mathcal{S}\}$ can be formalized as

$$Y(t) = \mathcal{G}_\theta(U)(t) = Q(v_L(t)), v_0(t) = P(U(t)), \text{ where each layer } v_l(t) \text{ is} \tag{13}$$

$$v_{l+1}(t) = \mathcal{I}_l(v_l)(t) = \sigma_{l+1}\left(W_l v_l(t) + \int_{\mathcal{T}} \kappa^{(l)}(t,s)v_l(s)ds + b_l(t)\right), l = 0, 1, \ldots, L-1 \tag{14}$$

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, $P \in \mathbb{R}^{v_0 \times \dim(\mathcal{S})}$ and $Q \in \mathbb{R}^{\dim(\mathcal{S}) \times v_L}$ are lifting and projection matrix, $\kappa^{(l)} \in C(\mathcal{T} \times \mathcal{T}; \mathbb{R}^{d_{v_{l+1}} \times d_{v_l}})$ is the kernel function for integration, and $b_l \in C(\mathcal{T}; \mathbb{R}^{d_{v_{l+1}}})$ is the bias function. And $P, Q, W_l, \kappa^{(l)}, b_l, l = 0, 1, \ldots, L-1$ are parameterized with neural networks θ.

B.2. Proof of Theorem 3.1

Theorem B.3 (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 $\forall t \in \mathcal{T}$*

$$(\mathcal{S}_{\phi,t} := \{Y \mid \phi(t, Y) \leq 0\} \subseteq \mathcal{S}_0) \bigwedge \left(\partial_Y \phi \cdot \frac{dY}{dt} + \partial_t \phi + \alpha\phi(t, Y) + C_{\alpha,T}\phi(0, U_0) \leq 0\right) \tag{15}$$

where $C_{\alpha,T} := \frac{\alpha}{e^{\alpha T}-1} > 0$ is a constant for finite-time convergence. Similarly, the boundary feasibility with neural BCBF $\phi(Y)$ holds if Equation (3) holds by letting $\partial_Y \phi = \nabla_Y \phi, \partial_t \phi = 0$.

Proof. To show the boundary feasibility of the boundary output of $Y(t)$ within user-specified safe set $\mathcal{S}_0$, by Definition B.1, we need to show

$$\exists t_0 \in [0, T], s.t. \forall t \in [t_0, T], Y(t) \in \mathcal{S}_0. \tag{16}$$

With the sublevel set $\mathcal{S}_{\phi,t}$ being the subset of $\mathcal{S}_0$, i.e., $\mathcal{S}_{\phi,t} := \{Y \mid \phi(t, Y) \leq 0\} \subseteq \mathcal{S}_0$, it is sufficient to prove

$$\exists t_0 \in [0, T], s.t. \forall t \in [t_0, T], \phi(t, Y(t)) \leq 0. \tag{17}$$

Now denote $\psi(t) := \phi(t, Y(t))$, by initial constant boundary condition $Y(0) = u(\mathbf{0}, 0) = u(\mathbf{1}, 0) = U_0$, we have the following equivalent inequalities hold,

$$\partial_Y \phi \cdot \frac{dY}{dt} + \partial_t \phi + \alpha\phi(t, Y) + C_{\alpha,T}\phi(0, Y(0)) \leq 0 \tag{18}$$

$$\iff \frac{d\phi(t, Y(t))}{dt} + \alpha\phi(t, Y) + C_{\alpha,T}\phi(0, Y(0)) \leq 0 \tag{19}$$

$$\iff \frac{d\psi(t)}{dt} + \alpha\psi(t) + C_{\alpha,T}\psi(0) \leq 0 \tag{20}$$

$$\iff e^{\alpha t}\frac{d\psi(t)}{dt} + e^{\alpha t}\alpha\psi(t) + e^{\alpha t}C_{\alpha,T}\psi(0) \leq 0, \forall t \in [0, T] \tag{21}$$

$$\iff \frac{d(e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t})}{dt} \leq 0 \tag{22}$$

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

$$[e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t}]|_{t=T} < [e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t}]|_{t=0} \tag{23}$$

$$\iff e^{\alpha T}\psi(T) + \frac{e^{\alpha T}}{e^{\alpha T}-1}\psi(0) < \psi(0) + \frac{1}{e^{\alpha T}-1}\psi(0) \tag{24}$$

$$\iff e^{\alpha T}\psi(T) < 0 \tag{25}$$

$$\iff \psi(T) < 0 \tag{26}$$

$$\iff \phi(T, Y(T)) < 0 \tag{27}$$

So at least at $t_0 = T$, $\phi(t_0, Y(t_0)) < 0$, which proves Equation (17) holds and the original theorem has been proved. Furthermore, let us look at the boundary feasible steps. Since $e^{\alpha t}\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}e^{\alpha t} = e^{\alpha t}(\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha})$ is non-increasing, with the strictly increasing and positive $e^{\alpha t}$, it is easy to find function $\psi(t) + \frac{C_{\alpha,T}\psi(0)}{\alpha}$ being non-increasing, i.e. $\psi(t)$ is non-increasing. Therefore, if $U_0 \leq 0$, $\phi(t, Y(t)) < \phi(0, Y(0)) = U_0 < 0, \forall t \in [0, T]$. If $U_0 > 0$, since MLP-ReLU parameterized neural BCBF ϕ and boundary control output Y are continuous, by mean value theorem, we have

$$\phi(0, Y(0)) > 0, \phi(T, Y(T)) < 0 \Rightarrow \exists t_0 \in [0, T], \phi(t_0, Y(t_0)) = 0. \tag{28}$$

Since $\psi(t) = \phi(t, Y(t))$ is non-increasing, we have

$$\exists t_0 \in [0, T], s.t. \forall t \in [t_0, T], \phi(t, Y(t)) \leq 0, \tag{29}$$

which concludes the proof. □

B.3. Proof of Theorem on Boundary Feasibility with Neural Operator

Theorem B.4 (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*

$$\partial_Y \phi(t, \mathcal{G}_\theta(U)) \frac{d\mathcal{G}_\theta(U)(t)}{dt} + \partial_t \phi(t, \mathcal{G}_\theta(U)) + \alpha\phi(t, \mathcal{G}_\theta(U)) + C_{\alpha,T}\phi(0, U(0)) \leq 0, \forall t \in \mathcal{T} \tag{30}$$

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

$$\frac{d\mathcal{G}_\theta(U)(t)}{dt} = \nabla Q^\top \prod_{l=0}^{L-1} \left(Diag(\sigma'_{L-l}) W_{L-1-l}\right) \nabla P^\top \frac{dU(t)}{dt} + \nabla Q^\top Diag(\sigma'_L) \sum_{i=0}^{L-1} \left([\prod_{j=1}^{i} W_{L-j} \right.$$
$$\left. Diag(\sigma'_{L-j})] \left(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-1-i)}(t, s)}{\partial t} v_{L-1-i}(s) ds + \frac{db_{L-1-i}(t)}{dt} \right) \right) = \Lambda_\theta(t)\dot{U}(t) + \mu_\theta(t) \tag{31}$$

Proof. The proof of Equation (30) is based on Theorem B.4 and Equation (31) can be derived by recursively applying Equation (9) to Equation (8). To show the boundary feasibility over sublevel set of ϕ hold, we first want to show Equation (31) holds. According to Definition B.2, we first rewrite the neural operator as

$$\begin{aligned} &Y(t) = \mathcal{G}_\theta(U)(t) = Q(v_L(t)), v_0(t) = P(U(t)), \text{ where each layer } v_l(t) \text{ is} \\ &v_{l+1}(t) = \mathcal{I}_l(v_l)(t) = \sigma_{l+1}\left(W_l v_l(t) + \int_{\mathcal{T}} \kappa^{(l)}(t, s) v_l(s) ds + b_l(t)\right), l = 0, 1, \ldots, L-1 \end{aligned} \tag{32}$$

where $P, Q, W_l, \kappa^{(l)}, b_l, l = 0, 1, \ldots, L-1$ are neural networks, kernel function $\kappa^{(l)}$, activation function σ_l and bias function b_l are first-order differential. Since the operator shares the same input function domain and output function domain over $t \in \mathbb{R}^+$, applying chain rule to Equation (32), we can find the derivative with respect to t for each layer as,

$$\frac{dY(t)}{dt} = \nabla Q^\top \frac{dv_L(t)}{dt}, \frac{v_0(t)}{dt} = \nabla P^\top \frac{dU(t)}{dt}, \text{ for each derivative } \frac{dv_{l+1}(t)}{dt}\ l = L-1, \ldots, 0, \tag{33}$$

$$\frac{dv_{l+1}(t)}{dt} = \mathcal{J}_l(\frac{dv_l}{dt})(t) = \text{Diag}(\sigma'_{l+1}) \left(W_l \frac{dv_l(t)}{dt} + \int_{\mathcal{T}} \frac{\partial \kappa^{(l)}(t, s)}{\partial t} v_l(s) ds + \frac{db_l(t)}{dt} \right) \tag{34}$$

Now put Equation (34) into Equation (33) recursively, we have

$$\frac{d\mathcal{G}(U)(t)}{dt} = \nabla Q^\top \frac{dv_L(t)}{dt} \tag{35}$$

$$=\nabla Q^\top \text{Diag}(\sigma_L')W_{L-1}\frac{dv_{L-1}(t)}{dt} + \nabla Q^\top \text{Diag}(\sigma_L')\left(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-1)}(t,s)}{\partial t} v_{L-1}(s)ds + \frac{db_{L-1}(t)}{dt}\right) \tag{36}$$

$$\begin{aligned}=&\nabla Q^\top \text{Diag}(\sigma_L')W_{L-1}\text{Diag}(\sigma_{L-1}')W_{L-2}\frac{dv_{L-2}(t)}{dt} + \nabla Q^\top \text{Diag}(\sigma_L')W_{L-1}\cdot \text{Diag}(\sigma_{L-1}')\cdot \\ &\left(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-2)}(t,s)}{\partial t} v_{L-2}(s)ds + \frac{db_{L-2}(t)}{dt}\right) + \nabla Q^\top \text{Diag}(\sigma_L')(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-1)}(t,s)}{\partial t} v_{L-1}(s)ds \\ &+ \frac{db_{L-1}(t)}{dt})\end{aligned} \tag{37}$$

$$\begin{aligned}=&\ldots \text{(recursively apply Equation (34))} \\ =&\nabla Q^\top \text{Diag}(\sigma_L')W_{L-1}\ldots \text{Diag}(\sigma_1')W_0\frac{dv_0(t)}{dt} + \nabla Q^\top \text{Diag}(\sigma_L')W_{L-1}\text{Diag}(\sigma_{L-1}')\cdots W_1 \\ &\text{Diag}(\sigma_1')\left(\int_{\mathcal{T}} \frac{\partial \kappa^{(0)}(t,s)}{\partial t} v_0(s)ds + \frac{db_0(t)}{dt}\right) + \cdots + \nabla Q^\top \text{Diag}(\sigma_L')W_{L-1}\cdot \text{Diag}(\sigma_{L-1}')\cdot \\ &\left(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-2)}(t,s)}{\partial t} v_{L-2}(s)ds + \frac{db_{L-2}(t)}{dt}\right) + \nabla Q^\top \text{Diag}(\sigma_L')(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-1)}(t,s)}{\partial t} v_{L-1}(s)ds \\ &\frac{db_{L-1}(t)}{dt})\end{aligned} \tag{38}$$

$$\begin{aligned}=&\nabla Q^\top \prod_{l=0}^{L-1}\left(\text{Diag}(\sigma_{L-l}')W_{L-1-l}\right)\nabla P^\top \frac{dU(t)}{dt} + \nabla Q^\top \text{Diag}(\sigma_L')\sum_{i=0}^{L-1}\Bigg(\left[\prod_{j=1}^{i} W_{L-j}\text{Diag}(\sigma_{L-j}')\right]\cdot \\ &\left(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-1-i)}(t,s)}{\partial t} v_{L-1-i}(s)ds + \frac{db_{L-1-i}(t)}{dt}\right)\Bigg)\end{aligned} \tag{39}$$

Note that the final expression in Equation (39) is actually linear with respect to $\dot{U}(t)$ and the weight and bias terms only depend on the parameters of the neural operator θ and the values at time t. Denote the linear weight and bias as $\Lambda_\theta(t), \mu_\theta(t)$

$$\Lambda_\theta(t) := \nabla Q^\top \prod_{l=0}^{L-1}\left(\text{Diag}(\sigma_{L-l}')W_{L-1-l}\right)\nabla P^\top, \mu_\theta(t) := \nabla Q^\top \text{Diag}(\sigma_L')\cdot \tag{40}$$

$$\sum_{i=0}^{L-1}\left(\left[\prod_{j=1}^{i} W_{L-j}\text{Diag}(\sigma_{L-j}')\right]\cdot\left(\int_{\mathcal{T}} \frac{\partial \kappa^{(L-1-i)}(t,s)}{\partial t} v_{L-1-i}(s)ds + \frac{db_{L-1-i}(t)}{dt}\right)\right), \tag{41}$$

then we have

$$\frac{dY(t)}{dt} = \frac{d\mathcal{G}(U)(t)}{dt} = \Lambda_\theta(t)\dot{U}(t) + \mu_\theta(t).$$

Since $Y(t) = \mathcal{G}(U)(t)$, Equation (30) is equivalent to

$$\partial_Y \phi \cdot \frac{dY}{dt} + \partial_t \phi + \alpha\phi(t, Y) + C_{\alpha,T}\phi(0, U(0)) \leq 0.$$

Similar to the proof of Theorem B.3, we have

$$\exists t_0 \in [0,T], s.t. \forall t \in [t_0, T], \phi(t, Y(t)) \leq 0, \tag{42}$$

which concludes the proof of boundary feasibility over the sublevel set of ϕ. □

Remark B.5. 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 B.4 is equivalent to Theorem 3.1. Then the boundary control input $U(t)$ satisfying Equation (30) is guaranteed to induce the boundary feasibility of boundary output $Y(t)$ within the user-specified safe set $\mathcal{S}_0$.

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)

C. 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 Appendix C.2 and the second one in Appendix C.3, following the setup of model training and evaluation metrics. Appendix D gives more details and results.

C.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 $Y(t) = u(0.5, t)$. 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 Appendix D.1. 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 (10) 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 2.1 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 feasible 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.

C.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

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

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.

C.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 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

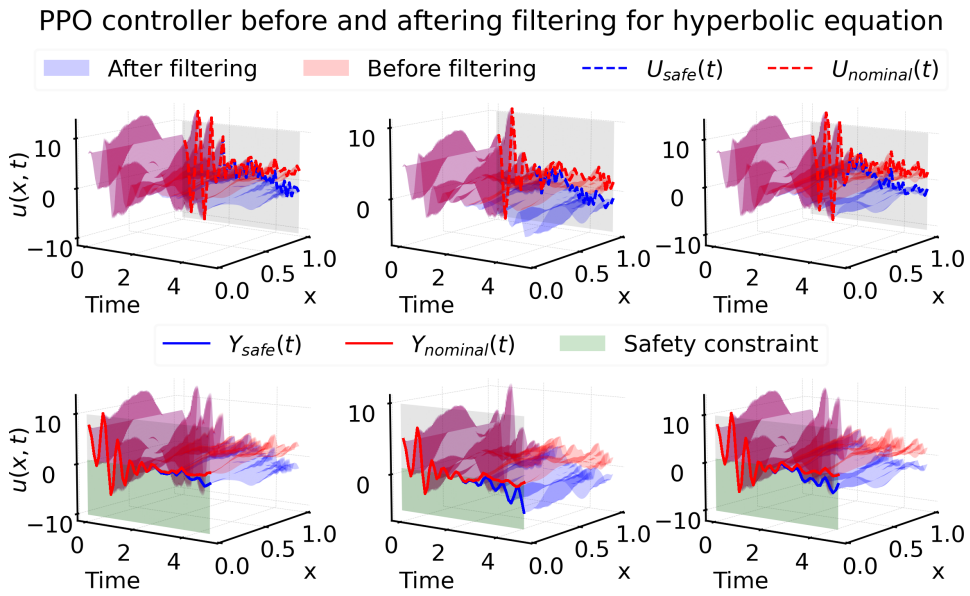

Figure 2: Visualization of three state trajectories $u(x,t)$ (left, mid, right) for hyperbolic equation under PPO controller with and without safety filtering. Boundary control inputs $U(t)$ are in dashed lines, and boundary outputs $Y(t)$ are in solid lines. The boundary constraint $Y(t) < 1$ is in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

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.

D. More Experiment Details and Results

D.1. Additional Experiment Setting

Data preparation. For 1D environments, the boundary input is $U(t) = u(1,t)$ while the boundary output for the hyperbolic PDE is $Y(t) = u(0,t)$ and the boundary output for the parabolic PDE $Y(t) = u(0.5,t)$ since $u(0,t) \equiv 0$. For the 2D environment, the boundary input is the x-axis consistent boundary condition, i.e., $u(x,1,t) \equiv U(x), v(x,1,t) \equiv 0, \forall x \in [0,1]$. The boundary output is $Y(t) = u(0.5,0.95,t), v(x,0.95,t) \equiv 0, \forall x \in [0,1]$, which has the maximum speed except for control input and can be viewed as an indicator for tracking performance. Note that we focus on the boundary output, which only depends on time in high-dimensional cases. The temporal resolution of collected trajectories is consistent with the control frequency of each environment in (Bhan et al., 2024), i.e., 50 steps in 5s for hyperbolic PDE, 1000 steps in 1s for parabolic PDE and 200 steps in 0.2s for Navier-Stokes PDE. We train the RL models PPO and SAC following the default hyper-parameters and unstable PDE settings in (Bhan et al., 2024) for hyperbolic and parabolic equations, while directly adopting the pre-trained models under default Navier-Stokes equation. For the data collection in the 1D hyperbolic equation, we evaluate the backstepping-based model (Krstic & Smyshlyaev, 2008a), PPO and SAC models with random initial conditions $U_0 \in [1,10]$ and collect 50k pairs of input and output $u(1,t), u(0,t)$ trajectories for each model. Similarly, for the 1D parabolic equation, we evaluate the backstepping-based model (Smyshlyaev & Krstic, 2004), PPO and SAC models with random initial conditions $U_0 \in [1,10]$ and collect 50k pairs of input and output $u(1,t), u(0.5,t)$ trajectories for each model. For the Navier-Stokes equation, we evaluate the model-based optimization method (Pyta et al., 2015), PPO and SAC models with random initial conditions $u_0 \in [-0.1,0.1]$ and default tracking ground truth and collect 10k pairs of input and output $u(0.05,1,t), u(0.5,0.95,t)$ trajectories for each model. After the data pairs are collected, we annotate the safety label with pre-defined safe constraints based on the original performance of each policy. We specify one-sided safe sets $\mathcal{S}_0 = \{Y : AY < b\}$ for stabilization tasks and two-sided safe sets $\mathcal{S}_0 = \{Y : |Y - Y_{gt}| < b\}$ for tracking tasks. Specifically, for the hyperbolic equation, $Y < 1$ for PPO and $Y < 0$ for SAC; for the parabolic equation, $Y < 0.6$ for PPO and $Y > -0.26$ for SAC; for the Navier-Stokes equation, $|Y - Y_{gt}| < 0.145$ for PPO and SAC models. Then we randomly split 90% as a training dataset and leave others as a test set. The safe reinforcement learning baselines are based

Algorithm 1 Safety Filtering Procedure for Discrete-time Implementation

Input: Initial and nominal control input $U^{\text{nominal}}_{0:M}$, neural operator $\mathcal{G}$, neural BCBF ϕ
Output: Filtered safe control input $U^{\text{safe}}_{1:M}$
Initialize $\Delta U^{\text{safe}}_{1:M} = \Delta U^{\text{nominal}}_{1:M} \leftarrow U^{\text{nominal}}_{1:M} - U^{\text{nominal}}_{0:M-1}, Y^{\text{predict}}_{1:M} \leftarrow \mathcal{G}(U^{\text{nominal}}_{1:M})$
for $m = 1 : M$ **do**
  Find $\Delta U^{\text{safe}}_{m}$ through QP in Equation (10) based on $\Delta U^{\text{nominal}}_{m}, Y^{\text{predict}}_{1:M}, \mathcal{G}, \phi, U^{\text{nominal}}_{0}$
  Update $U^{\text{safe}}_{1:M} \leftarrow \sum_{i=1}^{m} \Delta U^{\text{safe}}_{i} + U^{\text{nominal}}_{0}$
  Update $Y^{\text{predict}}_{1:M} \leftarrow \mathcal{G}(U^{\text{safe}}_{1:M})$
end for
return $U^{\text{safe}}_{1:M}$

on cumulative costs with empirical performance towards the safety constraints of the output boundary (Achiam et al., 2017; Ha et al., 2020; Liu et al., 2022b; 2024).

Model training. To train the neural operator models, we adopt the public package (NeuralOperators.jl), using the default gelu-activation model of FNO with channels of $(2, 64, 64, 64, 64, 64, 128, 1)$ and 16 modes, MNO with channels of $(2, 64, 64, 64, 64, 64, 1)$ and 16 modes. All the models are trained for 100 epochs with learning rate 10^{-3}, ℓ-2 regularization weight is 10^{-4}, ADAM optimizer and ℓ-2 loss. The resolutions and scales of hyperbolic, parabolic, and Navier-Stokes trajectories are 50, 1000, and 200 for 5s, 1s, and 0.2s, respectively. We keep the same setting for different environments and remark that we do not fully exploit the potential for the best performance of neural operators since it is not the main focus of this work. For the neural BCBF training, we directly use the finite difference of $Y(t)$ collected from real PDE dynamics instead of the gradient of the neural operator to avoid noise. Following the implementation of (Dawson et al., 2022; Zhang et al., 2023; Hu et al., 2024), we adopt 4-layer MLPs with ReLU with layer dimensions of (16,64,16,1) to model neural BCBFs. The time t is concatenated with $Y(t)$ as input for time-dependent neural BCBF $\phi(t, Y)$ while only $Y(t)$ is input for time-independent neural BCBF $\phi(t, Y)$. To construct the safe set loss in Equation (4), we adopt all the sampled steps along trajectories with unsafe labels while only choosing the "latest" safe sampled steps where boundary feasibility is satisfied in Definition 2.1, i.e. once $Y(t)$ is with safe label, it will never become unsafe in finite time T. For the boundary feasibility loss in Equation (7), due to too much data close to 0, we adopt a random drop of close-to-0 data to balance the output boundary data distribution. Specifically, for the hyperbolic equation, we keep 20% data within [-0.1,0.1] while keeping 20% data within [-0.01,0.01] for the parabolic equation. Following (Liu et al., 2022a), we adopt a regularization loss to avoid the shrinking of the sublevel set during training with a default weight of 1. We train all models with ADAM for 20 epochs with an initial learning rate of 0.01. The learning rate decay rate is 0.2 after each 4 epochs.

Discrete-time Implementation. 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 below, along with a discussion on how approximation errors affect safety filtering. We remark that iterative filtering with the prediction of $Y(t)$ at each step aims to avoid large approximation errors in Equation (31) in the discrete-time setting compared to one-time filtering for the whole trajectory. Besides, as the computation of QP is not yet real-time, it is not yet ready to interact with the real PDE dynamics. we adopt the predicted $Y(t)$ from the neural operator after each filtering step instead of real PDE dynamics. To handle the model mismatch issue between neural operator modeling and real underlying PDE dynamics, the filtering threshold $\eta > 0$ is introduced as a workaround and we leave the study of model mismatch of PDE dynamics as future work. Specifically, the safety filter is disabled when $\eta = 0$. The larger η is, the more boundary feasibility within the safe set will be achieved, showing a trade-off between stabilization and constraint satisfaction. The final control trajectory is found through Equation (43) with threshold $\eta = 2$ as default, mitigating the discrepancy between the PDE environment and the neural operator.

Table 5: Comparison time-independent and time-dependent safety filtering in different equations.

1D parabolic equation	Reward (mean±std) (starting at ∼0)	Feasible Rate (100 episodes)	Average Feasible Steps (1000 control steps)
PPO with filtering of $\phi(Y)$	162.9±19.6	0.46	**519.4**
PPO with filtering of $\phi(t, Y)$	168.2±23.5	**0.81**	**507.0**
SAC with filtering of $\phi(Y)$	157.9±6.9	**0.92**	**543.2**
PPO with filtering of $\phi(t, Y)$	**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 with filtering of $\phi(Y)$	-5.37±0.01	0.86	2.2
PPO with filtering of $\phi(t, Y)$	-5.72±0.17	**0.99**	**32.0**
SAC with filtering of $\phi(Y)$	-18.05±1.14	0.79	17.8
SAC with filtering of $\phi(t, Y)$	-18.36±1.25	**0.85**	**21.3**

$$U_{\text{safe}}(t) = \int_0^t \dot{U}(\tau)d\tau + U_{\text{nominal}}(0), \dot{U}(\tau) = \begin{cases} \dot{U}_{\text{safe}}(\tau), \text{if } \|\dot{U}_{\text{safe}}(\tau) - \dot{U}_{\text{nominal}}(\tau)\| \leq \eta, \\ \dot{U}_{\text{nominal}}(\tau), \text{ otherwise.} \end{cases} \tag{43}$$

We remark that iterative filtering with the prediction of $Y(t)$ at each step aims to avoid large approximation errors in Equation (31) in the discrete-time setting compared to one-time filtering for the whole trajectory.

D.2. Additional Results

Influence of filtering threshold. Since the boundary output $Y(t)$ is predicted from the neural operator in Algorithm 1, the model mismatch significantly influences the performance of safety filtering, where we handle it through a filtering threshold η in Equation (43). We investigate it to show the trade-off between general performance and boundary feasibility. From Figure 3, it can be seen that as the threshold goes up, the reward first slightly increases and then drops significantly, showing that the strong safety filtering may hurt the stability of the PPO controller due to the model mismatch between direct boundary mapping with the neural operator and underlying PDE dynamics. Besides, with a larger filtering threshold η, the average feasible steps become larger as the safety filtering becomes stronger, especially for time-dependent BCBF $\phi(t, Y)$, guaranteeing constraint satisfaction over boundary output. With small η, the average feasible steps may be less than the one without filtering because of more feasible trajectories with last-step feasibility. Safety filtering aligns with the stabilization to increase the reward, but the noise from the model mismatch between the neural operator and real dynamics will make the performance collapse if the safety filtering is too strong. For the boundary feasibility, we can see that the average feasible steps keep going up as the threshold increases, showing that the finite-time convergence is enhanced for the feasible trajectories. However, when the threshold becomes too large, e.g. $\eta = 10$, the feasible rate also decreases significantly because the system is no longer stable, as the reward indicates.

Comparison of asymptotic and finite-time boundary feasibility. In Table 6, we show the comparison of safety filtering with BCBF $\phi(t, Y)$ for 1D hyperbolic equation for asymptotic and finite-time boundary feasibility. Asymptotic boundary feasibility is with the neural BCBF trained and tested with $C_{\alpha,T} = \lim_{T\to\infty} \frac{\alpha}{e^{\alpha T}-1} = 0$ while finite-time boundary feasibility is with $C_{\alpha,T} = 0.02$ using $T = 50$. It can be seen that BCBF with finite-time feasibility has a better feasible rate, especially the SAC model, as asymptotic feasibility is weaker than finite-time feasibility and takes longer steps to converge. However, for the general reward performance, since asymptotic feasibility causes weaker filtering effects, the reward tends to be closer to the vanilla reward without filtering compared to finite-time feasibility, which is validated in Table 6.

More comparison with different operators. In this section, we show the comparison of two neural operators, FNO (Li et al., 2020a) and MNO (Li et al., 2022) for the safety filter performance with $\phi(Y)$ in learning the boundary mapping from

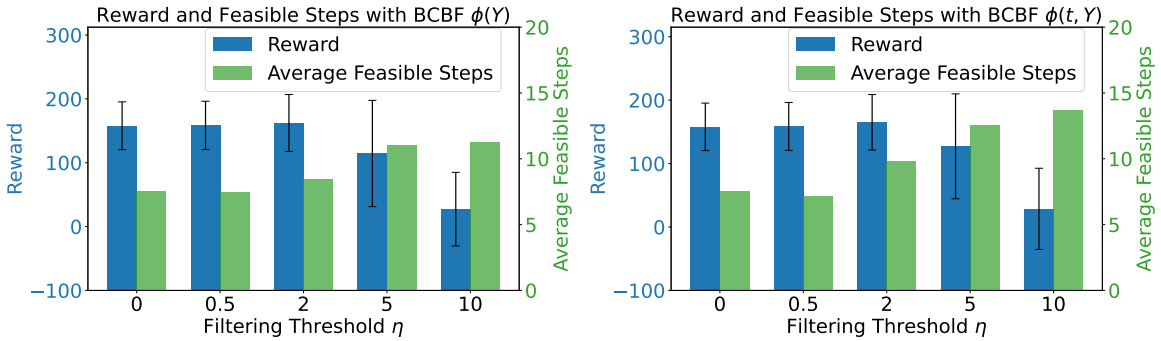

Figure 3: The reward and feasible rate under different filtering threshold η in Equation (43) with BCBF $\phi(Y)$ (left) and $\phi(t, Y)$ (right) for PPO model in hyperbolic equation. Note that $\eta = 0$ indicates the vanilla PPO model without safety filtering.

Table 6: Results of filtering with BCBF $\phi(t, Y)$ for 1D hyperbolic equation for asymptotic $C_{\alpha,T} = \lim_{T\to\infty} \frac{\alpha}{e^{\alpha T}-1} = 0$ and finite-time $C_{\alpha,T} = \frac{\alpha}{e^{\alpha T}-1} = 0.02$ at $T = 50, \alpha = 10^{-5}$.

Different neural operators	Reward (mean±std) (starting at ∼-300)	Feasible Rate (100 episodes)	Average Feasible Steps (50 control steps)
PPO for asymptotic feasibility	163.8±40.6	0.70	8.1
PPO for finite-time feasibility	**165.0**±43.7	**0.71**	**9.8**
SAC for asymptotic feasibility	**104.6**±98.6	0.56	**14.7**
SAC for finite-time feasibility	103.4±96.4	**0.85**	13.9

control input $U(t)$ to output $Y(t)$ for 1D hyperbolic equation. Note that MNO models have larger model complexity than FNO models. Different from Table 4, in Table 7, we can see that with weaker BCBF $\phi(Y)$, MNO performs no worse than FNO in feasible rate and reward, showing that larger model complexity will compensate the performance of BCBF in the safety filter framework.

More visualization of hyperbolic and Navier-Stokes equations. Here, we visualize the trajectories under 1D hyperbolic equation using a SAC controller without and with safety filtering of $\phi(t, Y)$, as shown in Figure 4. Similar to 2, for each trajectory, the state value $u(x, t)$ after filtering is lower than that before filtering, i.e., the blue area is lower than the red area. For the output boundary, the filtered one $Y(t)_{\text{safe}}$ in blue solid lines goes towards the constraint $Y(t) < 0$ compared to the nominal boundary output $Y(t)_{\text{nominal}}$ in red solid lines, because of the output boundary. The difference is not very large in the last two figures because the threshold is relatively small to keep the stability of the output. As the visualization shows in Figure 5, it can be seen that the mid-upper high-speed tracking performance is improved compared to the baseline without filtering due to the constraint satisfaction. However, since the output boundary is just one point in the high-speed part, the general performance after filtering is not improved significantly, which is consistent with the findings in Table 1.

Table 7: Results of filtering with BCBF $\phi(Y)$ under different neural operator modeling for first-order transport equation. The boundary feasibility constraint is $Y < 1$ for PPO and $Y < 0$ for SAC models.

Filtering with different BCBFs	Reward (mean±std) (starting at ∼-300)	Feasible Rate (100 episodes)	Average Feasible Steps (50 control steps)
PPO w. MNO	**162.9**±45.2	**0.68**	**8.7**
PPO w. FNO	162.3±44.5	0.63	8.3
SAC w. MNO	103.2±98.3	**0.59**	15.4
SAC w. FNO	**103.3**±98.4	0.57	**15.7**

Table 8: Comparison of before QP and after QP filtering with different thresholds using $\phi(Y)$ and $\phi(t, Y)$ for PPO model under hyperbolic equation.

Filtering with $\phi(Y)$	Reward (mean±std)	Feasible Rate	Average Feasible Steps
Before QP (baseline)	157.90±37.46	0.63	7.56
After QP with threshold 0.5	158.45±37.82	0.65	7.49
After QP with threshold 2	162.26±44.53	0.63	8.49
After QP with threshold 5	114.40±83.25	0.67	11.01
After QP with threshold 10	27.28±57.62	0.57	11.30
Filtering with $\phi(t, Y)$	Reward (mean±std)	Feasible Rate	Average Feasible Steps
Before QP (baseline)	157.90±37.46	0.63	7.56
After QP with threshold 0.5	158.60±37.76	0.68	7.19
After QP with threshold 2	165.04±43.73	0.71	9.80
After QP with threshold 5	127.18±82.67	0.73	12.60
After QP with threshold 10	28.61±64.03	0.57	13.74

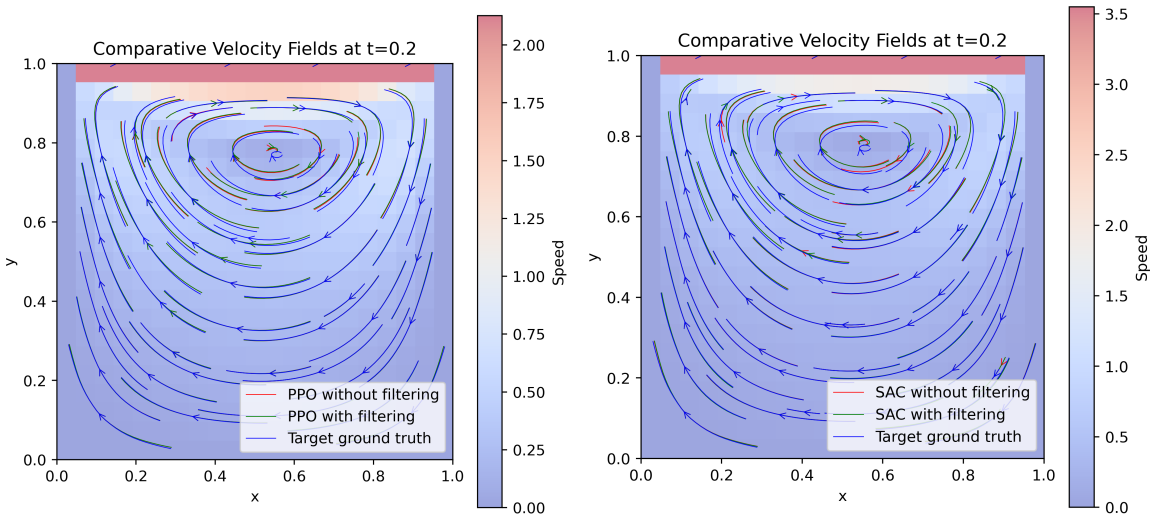

Figure 5: Visualization of tracking performance with PPO and SAC models before and after filtering with $\phi(t, Y)$ at the end time step of the trajectory for Navier-Stokes equation.

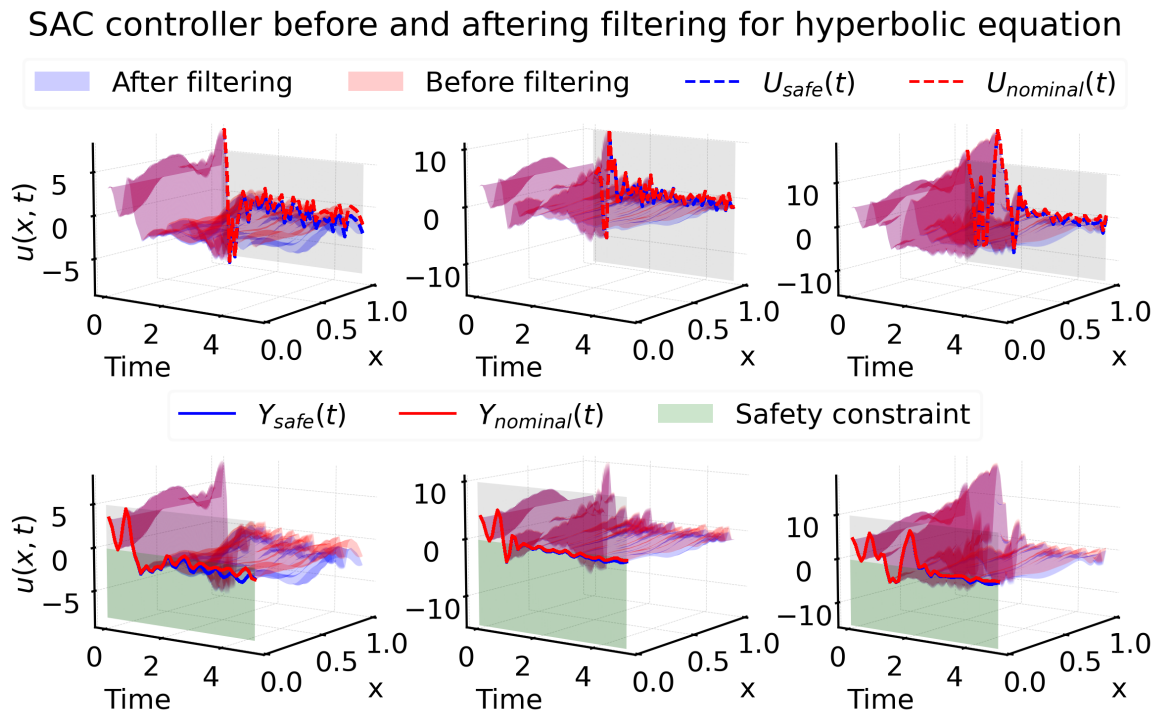

Figure 4: Visualization of state $u(x,t)$ of hyperbolic equation under SAC controller with (in blue) and without (in red) filtering. Boundary control inputs $U(t)$ are in dashed lines and boundary output $Y(t)$ are in solid lines. The boundary constraint $Y(t) < 0$ is in green.

E. Conclusion and Limitation

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.

Since the proposed method is based on neural operator modeling instead of real PDE dynamics, it does not directly solve the problem of model mismatch which may hurt the safety filtering performance in the implementation. We mark this important point as future work. Also, for PDE dynamics with higher-dimensional states, future work is needed to investigate how BCBF can deal with spatially dependent boundaries under complicated boundary constraint settings and safe sets. Another limitation lies in that we do not adopt online safety filtering under the real PDE dynamics, which can be further explored by replacing the offline filtered control input trajectory with the real-time safety filtering at each step in Algorithm 1 in the real-world applications. It is also interesting to omit the iterative filtering by prediction using the one-time filtering for the whole trajectory based on Equation (30), which has the challenge of the nonlinear dependence of the neural operator derivative at the initial time. More work can also be explored using neural network verification (Wei et al., 2024b; Yang et al., 2024) to ensure the safety and robustness under input perturbation (Cheng et al., 2024; Liu et al., 2023).