Proceedings of Machine Learning Research vol 242:1–26, 2024 6th Annual Conference on Learning for Dynamics and Control

PDE Control Gym: A Benchmark for Data-Driven Boundary Control of Partial Differential Equations

Luke Bhan[*] LBHAN@UCSD.EDU
University of California, San Diego
Yuexin Bian[*] YUBIAN@UCSD.EDU
University of California, San Diego
Miroslav Krstic KRSTIC@UCSD.EDU
University of California, San Diego
Yuanyuan Shi YYSHI@UCSD.EDU
University of California, San Diego

Editors: A. Abate, K. Margellos, A. Papachristodoulou

Abstract

Over the last decade, data-driven methods have surged in popularity, emerging as valuable tools for control theory. As such, neural network approximations of control feedback laws, system dynamics, and even Lyapunov functions have attracted growing attention. With the ascent of learning based control, the need for accurate, fast, and easy-to-use benchmarks has increased. In this work, we present the first learning-based environment for boundary control of PDEs. In our benchmark, we introduce three foundational PDE problems - a 1D transport PDE, a 1D reaction-diffusion PDE, and a 2D Navier–Stokes PDE - whose solvers are bundled in an user-friendly reinforcement learning gym. With this gym, we then present the first set of model-free, reinforcement learning algorithms for solving this series of benchmark problems, achieving stability, although at a higher cost compared to model-based PDE backstepping. With the set of benchmark environments and detailed examples, this work significantly lowers the barrier to entry for learning-based PDE control - a topic largely unexplored by the data-driven control community. The entire benchmark is available on Github along with detailed documentation and the presented reinforcement learning models are open sourced.

Keywords: Partial Differential Equation Control, Nonlinear Systems, Benchmarking for Data-Driven Control, Reinforcement Learning

1. Introduction

As learning-based control has exploded across both academia and industry, the need for fast, and accurate bench-marking is heightened. For example, perhaps the most visible impact of proper bench-marking is in the field of computer-vision resulting in 15-years of breakthrough results from AlexNet Krizhevsky et al. (2012) to neural radiance fields (NeRFs) Mildenhall et al. (2020). Despite this, the control community has, justifiably, forgone consistent efforts in bench-marking as the community spawned from an applied mathematics perspective where the focus was behind *provable* stability guarantees. However, given the recent exploration surrounding data-driven control methods Berberich et al. (2023); Feng et al. (2023), designing fast, well-documented, and challenging benchmarks is of utmost importance to ensure new learning-based control approaches are consistently advancing the state of the art.

[*] equal contribution

In this work, we develop a benchmarking suite for learning-based boundary control of PDEs. Boundary control of PDEs is of elevated importance compared to control across the full domain as many real world problems *cannot* control the PDE across the entire domain, but only at the boundary input. Thus, boundary control is physically more realistic as the actuation and sensing are generally non-intrusive Krstic and Smyshlyaev (2008b). For example, in fluid flows, the engineer only gets control access to the surrounding walls containing the fluid Di Meglio et al. (2012) or in temperature manufacturing Bian et al. (2024), the engineer is typically unable to set the temperature of entire plate, but only a specific edge. Furthermore, boundary control is extremely powerful in modeling macro-level traffic congestion Huan Yu (2023) as modern highways typically only enable actuation and sensing at the on/off ramps. Lastly, we briefly mention that boundary control is one of the only approaches to handle chemical and combustion processes Izadi et al. (2015) and moving boundary problems such as the Stefan Problem with applications to both 3D printing Koga and Krstic (2020) and the excitation of neuron growth for abating neurological diseases such as Parkinson's and Alzheimer's Demir et al. (2024). However, for a majority of these applications, each researcher typically develops their own simulations and thus there is no standard library with a universal set of problems to test new algorithms. Furthermore, for even the standard model-based algorithms such as PDE backstepping, it is challenging to implement the control algorithm as this typically requires the solution of a *separate* Goursat PDE Vazquez et al. (2023). Thus, in this work, we introduce the first library containing a set of general PDE control problems and implementations of their corresponding model-based control algorithms that can be easily modified to fit the wide array of aforementioned target applications.

Contributions This paper has three main contributions. First, we introduce, design, and formalize the first benchmark suite for PDE control including 3 classical problems ranging from boundary stabilization for 1D transport (hyperbolic) and reaction-diffusion (parabolic) PDEs to trajectory following for the 2D Navier-Stokes PDEs. Along with the proposal of the PDE control benchmarking suite, we parameterize the numerical scheme implementations as RL gyms - effectively decoupling the PDE solvers from the controller design, enabling the use of *any pre-implemented learning algorithm* for PDE control. Second, utilizing our benchmark suite, we train the first set of *model-free* RL controllers which effectively stabilize hyperbolic and parabolic PDE problems and achieve effective tracking for the 2D Navier-Stokes equations. We then compare the resulting controllers to classical algorithms such as PDE backstepping and adjoint-based optimization highlighting the trade-offs between performance. Lastly, we provide extensive documentation and numerous examples for the training of RL controllers, implementation of classical control algorithms and of course the integration of new PDE control problems into the benchmark suite.

2. Related Work

2.1. Learning-based PDE benchmarks

Currently, to the author's knowledge, there are no benchmarking suites focused on *boundary control* of PDEs as most benchmarks such as Takamoto et al. (2022); Gupta and Brandstetter (2022) present datasets for learning PDE solution maps from initial conditions. These benchmarks are typically used for comparing neural network-based PDE solvers like neural operators Lu et al. (2021); Li et al. (2021) and PINNs Raissi et al. (2019). Although these benchmarks are effective, they do not allow users to incorporate boundary control or alter the PDEs within their datasets. The

Benchmarking for machine learning in PDEs	Compilation of a premade dataset	Supports control	Differentiable PDE solver	Supports custom PDEs	Supports reinforcement learning	Implementation of model-based control
PhiFlow Holl et al. (2020)	✗	✓	✓	✓	✗	✗
PDEBench Takamoto et al. (2022) (created from PhiFlow)	✓	✗	✗	✗	✗	✗
PDEArena Gupta and Brandstetter (2022) (created from PhiFlow)	✓	✗	✗	✗	✗	✗
PDE Control Gym (Ours)	✗	✓	✗	✓	✓	✓

Table 1: Comparison of benchmarks for machine learning in PDEs.

ϕ_{Flow} framework Holl et al. (2020) provides a flexible PDE solver that supports automatic differentiation for the calculation PDE derivatives to be used in control algorithms. While adaptable for PDE control, it does not natively support RL algorithms and lacks a set of model-based controllers for learning-based control comparisons. Lastly, it is worth noting that while there are ODE control suites like Duan et al. (2016); Tassa et al. (2018) tailored for ODE control tasks. Lastly, we mention a concurrent work that also integrates **in-domain** PDE control into a RL library Zhang et al. (2024); however, the class of problems they focus on - namely in-domain control - is different from the boundary control problems we benchmark in this gym. Thus, the PDE control gym, to our knowledge, represents the *first* PDE-focused benchmarking suite for learning-based boundary-control algorithms.

2.2. Learning as a tool for PDE control

As with most scientific disciplines, machine learning has had a broad impact in PDE control. In 1D PDE problems, a series of work has been developed to use neural operators for approximating control feedback laws Bhan et al. (2023a,b); Krstic et al. (2024); Qi et al. (2023), under a supervised learning framework with provable stability guarantees. Furthermore, an optimal control approach using PINNs is explored in Mowlavi and Nabi (2023). Additionally, the closest paper to this work is by Yu et al. (2022) who presented the first exploration utilizing RL for PDE boundary control. However, they do not explore the benchmark PDE problems presented in this paper instead focusing on Aw-Rascale-Zhang (ARZ) traffic model.

2.3. Reinforcement learning in control

Reinforcement learning (RL) has demonstrated significant success in various control applications, including robotics Brunke et al. (2022), power systems Chen et al. (2022), and autonomous driving Kiran et al. (2021). From a controls perspective, deep RL (DRL) algorithms learn feedback laws that maps observations (states) into actions (control inputs), typically via neural networks (NN). These RL controllers are trained to optimize specific reward functions, such as the L^2 spatial norm of states for stabilization tasks Yu et al. (2022). The most appealing feature of deep RL is its *model-free* nature, allowing it to control complex systems without requiring explicit model estimations. Consequently, RL has potential to outperform model-based control methods in highly complex tasks with hard-to-model dynamics. To demonstrate the use of PDE Control Gym, we conduct experiments using off-the-shelf RL algorithms implemented with Stable-Baselines3 Raffin et al. (2021). We selected the off-policy soft actor-critic (SAC) (Haarnoja et al., 2018a) and on-policy proximal policy optimization (PPO) (Schulman et al., 2017) algorithms for their demonstrated efficiency in solving challenging continuous control tasks Duan et al. (2016).

3. Formalization of PDE Control Problems

3.1. General PDE control problem

We consider a partial differential equation (PDE) defined on a domain $\mathcal{X}$, which can be either one-dimensional (1D), $\mathcal{X} = [0, 1] \subset \mathbb{R}$, or two-dimensional (2D), $\mathcal{X} = [0, 1] \times [0, 1] \subset \mathbb{R}^2$. The time domain is $\mathcal{T} = [0, T] \subset \mathbb{R}^+$. Let $u(x, t), x \in \mathcal{X}, t \in \mathcal{T}$ describe the state of the system governed by the PDE according to the dynamics

$$\frac{\partial u}{\partial t} = \mathcal{P}\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots, U(t)\right), \tag{1}$$

where $\mathcal{P}$ is the partial differential equation(s) that model(s) the system dynamics, and $U(t)$ is the control function. Then, the goal of a PDE control problem is to optimize a cost function (e.g. regulate $u(x, t)$ to be the desired trajectory while reducing the control cost, stabilize the PDEs) from just boundary inputs. Note that in some methods such as PDE backstepping, optimization is forgone in favor of just asymptotic stabilization as the infinite dimensional nature of PDE control is extremely challenging.

3.2. Markov decision processes (MDPs) for PDE control

We give a brief overview of the components of the MDP governing both the 1D and 2D support problems. More details about the specific MDPs governing the examples in Section 5 can be found in the Appendix A.2 and B.2.

	1D Hyperbolic		**1D Parabolic**		**2D Navier-Stokes**	
Supported Configuration	**Sensing** ($o(t)$)	**Actuation** ($a(t)$)	**Sensing** ($o(t)$)	**Actuation** ($a(t)$)	**Sensing** ($o(t)$)	**Actuation** ($a(t)$)
full-state	$u(x,t)$	$u(1,t)$	$u(x,t)$	$u(1,t)$	$u(x,y,t)$	$u(x,1,t)$
colloacted	$u_x(1,t)$	$u(1,t)$	$u_x(1,t)$	$u(1,t)$	—	—
anti collocated	$u(0,t)$	$u(1,t)$	—	—	—	—
anti collocated	$u_x(0,t)$	$u(1,t)$	$u_x(0,t)$	$u(1,t)$	—	—
full-state	$u(x,t)$	$u_x(1,t)$	$u(x,t)$	$u_x(1,t)$	$u(x,y,t)$	$u(x,0,t)$
collocated	$u(1,t)$	$u_x(1,t)$	$u(1,t)$	$u_x(1,t)$	—	—
anti collocated	$u(0,t)$	$u_x(1,t)$	—	—	—	—
anti collocated	$u_x(0,t)$	$u_x(1,t)$	$u_x(0,t)$	$u_x(1,t)$	—	—
full-state	—	—	—	—	$u(x,y,t)$	$u(1,y,t)$
full-state	—	—	—	—	$u(x,y,t)$	$u(0,y,t)$

Table 2: Configurations for actuation and sensing supported by the PDE Control Gym for the three problems. Full state indicates the measurement is the entire PDE state, collated indicates that the sensing and measurement is done at the same boundary point, and anti-collocated indicates sensing and measurement are done at opposite boundary points. The configurations marked in blue correspond to the experiment examples in Section 5.

State and Observation Space. In all PDE control problems addressed, the state $s(t)$ at time t is represented by the PDE value $u(x,t), x \in \mathcal{X}$. To enhance flexibility for different PDE tasks such as observer design, we have developed different partial state measurement settings, which we denote as the observation space $o(t)$. The types of sensing supported for each problem are detailed in Table 3.2. Furthermore, we offer the ability to introduce custom noise functions (with built-in support for Gaussian noise). This allows users to simulate real-world sensor noise in their experiments.

Action Space. The action $a(t) = U(t)$ for both the RL and control agents is determined by the actuation locations and boundary condition type. In 1D hyperbolic and parabolic systems, we consider both Neumann and Dirichlet boundary actuation $U(t) \in \mathbb{R}$ at either boundary, with four possible cases: 1) $u(0,t) = U(t)$, 2) $u(1,t) = U(t)$, 3) $u_x(0,t) = U(t)$, and 4) $u_x(1,t) = U(t)$. Considering the symmetry between the boundaries $x = 0$ and $x = 1$, there are eight distinct combinations for the 1D Hyperbolic PDE problem and six for the 1D Parabolic PDE problem, including an additional boundary condition at $u(0,t)$. These combinations are outlined in the first two sections of Table 3.2. For the 2D Navier–Stokes problem, we consider Dirichlet-type boundary actuation on any of the four boundaries: top, bottom, left, and right. The gym also allows users to customize actuator positions, enabling research into optimizing both location and actuation type.

State Evolution. To simulate the PDE system evolution with the control input $U(t)$, we use a first-order Taylor approximation for temporal evolution,

$$u(t+\Delta t) = u(t) + \Delta t \cdot \mathcal{P}\left(u(t), \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots, U(t)\right) . \tag{2}$$

Spatial derivatives are approximated using appropriate finite difference schemes and are explicitly given for each gym environment in the Appendix A.1 and B.1. In practice, selecting the time step Δt and spatial discretization Δx for each problem requires careful consideration, particularly based on the number of approximated spatial derivatives. Nonetheless, we found that reasonable choices such as $\Delta x = 0.01$, $\Delta t = 0.0001$ yield both fast and numerically stable results.

Reward. Reward shaping plays a pivotal role in the training of RL algorithms. Generally speaking, for stabilization tasks, it is appropriate to employ a form of trajectory-based reward

$$\int_0^T \left(\int_{x \in \mathcal{X}} \|u(x,t)\|^2 dx + \|U(t)\|^2 \right) dt \,, \tag{3}$$

which minimizes the state magnitudes and control efforts. For tracking tasks, a trajectory reward as

$$\int_0^T \left(\int_{x \in \mathcal{X}} \|u(x,t) - u_{ref}(x,t)\|^2 dx + \|U(t) - U_{ref}(t)\|^2 \right) dt \,, \tag{4}$$

is a reasonable choice as it penalizes deviations from the reference trajectory given in both state $u_{ref}(x,t)$ and control actions $U_{ref}(t)$. However, in practice, for the 1D hyperbolic and parabolic PDE problems, we found that the reward as given in (3) was insufficient for training and thus we use a specifically tuned reward that penalizes the difference of the L_2 norms between the current state and next state after action $a(t)$ (presented in Appendix A.2 and B.2).

4. Benchmark PDE Control Tasks

4.1. 1D Hyperbolic (transport) PDEs

We consider the benchmark transport PDE in the form

$$u_t(x,t) \quad = \quad u_x(x,t) + \beta(x)u(0,t), \tag{5}$$

for $x \in [0,1), t \in [0,T]$. Physically, (5) is a "transport process (from $x = 1$ towards $x = 0$) with recirculation" of the outlet variable $u(0,t)$. Recirculation causes *instability* and the goal is to stabilize "full-state" recirculation from only boundary inputs. In practice, we consider the same PDE as studied in Bhan et al. (2023a) where β is governed by the Chebyshev polynomial $\beta(x) = 5\cos(\gamma\cos^{-1}(x))$ and Dirichlet actuation $u(1,t) = U(t)$. Classically, this PDE, with recirculation, has been a seminal-benchmark for PDE backstepping, as the 1D transport PDE can model a wide range of applications from chemical processes to shallow water waves and traffic flows Krstic and Smyshlyaev (2008a).

Model-based backstepping control. The backstepping controller given by the following

$$U(t) \quad = \quad \int_0^1 k(1-y)u(y,t)dy\,, \tag{6}$$

$$k(x) \quad = \quad -\beta(x) + \int_0^x \beta(x-y)k(y)dy\,, \tag{7}$$

for $x \in [0,1]$. (6) results in stabilization of (5) Krstic and Smyshlyaev (2008a). In practice, the backstepping kernel (7) is implemented using the successive approximations approach although a Laplace transform approach is also viable Krstic and Smyshlyaev (2008b).

4.2. 1D Parabolic (reaction-diffusion) PDEs

We consider the benchmark reaction-diffusion PDEs governed by recirculation function $\lambda(x)$ as

$$u_t(x,t) \quad = \quad u_{xx}(x,t) + \lambda(x)u(x,t), \tag{8}$$

$$u(0,t) \quad = \quad 0, \tag{9}$$

with Dirichlet or Neumann actuation at $x = 1$. Again, instability is caused by the $\lambda(x)u(x,t)$ term otherwise the problem would simplify to the classical heat equation. This PDE appears in different applications ranging from a chemical tubular reactor Shi et al. (2022) to electro-chemical battery models Moura et al. (2014) and diffusion in social networks Wang et al. (2020).

Model-based backstepping control. For the PDE (8), (9), with Dirichlet boundary actuation $u(1,t) = U(t)$, the backstepping controller with full state measurement is given by the following Smyshlyaev and Krstic (2004, 2010),

$$U(t) = \int_0^1 k(1,y)u(y,t)dy. \tag{10}$$

where $k(x,y) \in C^2(\tilde{\mathcal{T}}), \tilde{\mathcal{T}} = \{0 \leq y \leq x \leq 1\}$.

$$k_{xx}(x,y) - k_{yy}(x,y) \quad = \quad \lambda(y)k(x,y), \quad \forall (x,y) \in \breve{\mathcal{T}}\,, \tag{11}$$

$$
\begin{aligned}
k(x,0) &= 0\,, && (12)\\
k(x,x) &= -\frac{1}{2}\int_0^x \lambda(y)dy\,, && (13)
\end{aligned}
$$

where $\breve{\mathcal{T}} = \{0 < y \leq x < 1\}$.

4.3. 2D Navier-Stokes PDEs

We consider the 2D in-compressible Navier-Stokes equations as the third benchmark control task,

$$
\begin{aligned}
&\nabla \cdot \boldsymbol{u} = 0\,, && (14a)\\
&\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} \qquad . && (14b)
\end{aligned}
$$

With slight abuse of notation, we denote the spatial variable (in 2D) as $\boldsymbol{x} = (x, y) \in \mathcal{X} = [0, 1] \times [0, 1]$, and $\boldsymbol{u} = (u, v) : \mathcal{X} \times \mathcal{T} \to \mathbb{R}^2$ represents 2D velocity field, ν is the kinematic viscosity of the fluid, ρ is the fluid density, and p is the pressure field. Navier-Stokes equation is fundamental in fluid dynamics with extensive applications including aerodynamic design, pollution modeling, and wind turbine flows Jameson et al. (1998); Li et al. (2021). For the experiments, we consider boundary control along the top boundary as $\boldsymbol{u}(x, 0, t) = U(x, t), \forall x \in [0, 1]$. All other boundary conditions are Dirichlet boundary conditions where the velocity is set to be 0. The task here is to find boundary control $U(x, t)$ such that the resulting velocity field is close to the reference trajectory given in both desired velocity field $\boldsymbol{u}_{ref}(\boldsymbol{x}, t)$ and desired actions $U_{ref}(x, t)$.

Model-based optimization-based control. We provide an optimization-based controller based on Pyta et al. (2015) as the model-based control baseline.

$$
\begin{aligned}
\min_{U(x,t)} J(U(\cdot,t), \boldsymbol{u}) \quad &= \quad \frac{1}{2}\int_{\mathcal{T}}\int_{\mathcal{X}} \|\boldsymbol{u}(\boldsymbol{x},t) - \boldsymbol{u}_{ref}(\boldsymbol{x},t)\|^2 \mathrm{d}\boldsymbol{x}\mathrm{d}t \\
&\quad + \frac{\gamma}{2}\int_{\mathcal{T}} \|U(\cdot,t) - U_{ref}(\cdot,t)\|^2 \mathrm{d}t && (15a)\\
\text{s.t.} \qquad & (14a), (14b), \qquad \boldsymbol{u}(x,0,t) = U(x,t), \forall x \in [0,1]. && (15b)
\end{aligned}
$$

To ensure computational tractability, the control actions are set to be the tangential, uniform velocity, i.e., $u(x, 0, t) = U(t), v(x, 0, t) = 0$, followed in Pyta et al. (2015). The optimal control actions are obtained by solving the PDE-constrained optimization problem presented in (15). This solution employs Lagrange multipliers, utilizing the adjoint method McNamara et al. (2004); Gunzburger (2002) for gradient computation of the Lagrangian function.

5. Experiments

For each of the 3 environments in PDE Control Gym, we implemented baseline model-based control algorithms as well as off-the-shelf RL algorithms including soft actor-critic (SAC) (Haarnoja et al., 2018a) and proximal policy optimization (PPO) (Schulman et al., 2017) trained using Stable-Baselines3 (Parameters available in Appendix A.2 and B.2). We note that all the experiments can be trained in under 1 hour (Nvidia RTX 3090ti) and entire trajectories can be simulated in seconds.

Algorithm	**Hyperbolic PDE** **Average Episode Reward for Trained Policy** ↑	**Parabolic PDE** **Average Episode Reward for Trained Policy** ↑	**Navier-Stokes PDE** **Average Episode Reward for Trained Policy** ↑
Model-based	**246.3**	**299.1**	-7.931
PPO	172.3	293.3	**-5.370**
SAC	184.2	229.1	-17.829

Table 3: Resulting control algorithm performance on 50 test episodes in each gym (larger value indicates better performance). The model-based algorithm for the hyperbolic and parabolic PDEs are the backstepping schemes given in (6), (7) and (10), (11), (12), (13) respectively while the method for the Navier-Stokes PDE solves the optimization problem (15).

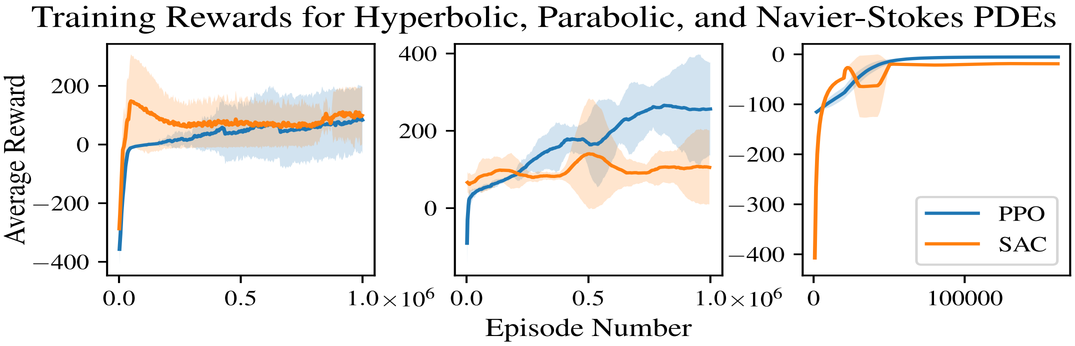

Figure 1: Rewards for training PPO (blue) and SAC (orange) on the 1D transport PDE, 1D reaction-diffusion PDE, and 2D Navier-Stokes PDE from left to right. The solid lines represent the mean and the shaded bounds are 95% confidence intervals across 5 seeds.

5.1. 1D Hyperbolic (transport) PDEs

Experimental design Our experimental setup for the Hyperbolic 1D problem, detailed in Section 4.1, considers full state measurements and boundary control $u(1,t) = U(t)$. We use the Chebyshev polynomial recirculation function $\beta(x) = 5\cos(\gamma\cos^{-1}(x))$ from Bhan et al. (2023a), with $\gamma = 7.35$ (future studies may vary γ). Each episode is initiated from a random initial condition, $u(x,0) \sim$ Uniform(1, 10). This setup presents a challenging control scenario, as the open loop system ($U(t) = 0$) is unstable (See Figure 5 in Appendix).

Results We now present detailed results on the policies trained and their comparison with model-based backstepping. In the left of Figure 1, we present the average reward functions for both RL algorithms over 1 million training steps. Then, in Table 3, we present the average reward where we run the trained final RL policies, and the model-based backstepping policy for 50 test episodes with different initial conditions, noting that model-based backstepping performs the best. Additionally, in Figures 2 we provide a comparison across all 3 control approaches where $u(x,0) = 10$. We can clearly see that although all 3 policies are stabilizing for the examples, model-based PDE back-

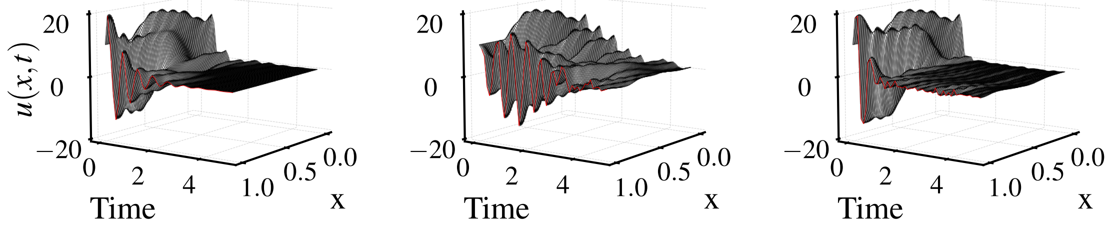

Figure 2: Example of the 1D transport PDE system stabilization using backstepping, PPO, and SAC (left to right) under initial conditions $u(x, 0) = 10$. The recirculation coefficient is defined as $\beta(x) = 5 \cos(\gamma \cos^{-1}(x))$ with $\gamma = 7.35$.

stepping again performs the best and the RL control signals are high oscillatory leaving room for improvement in applying model-free PDE control algorithms.

5.2. 1D Parabolic (reaction-diffusion) PDEs

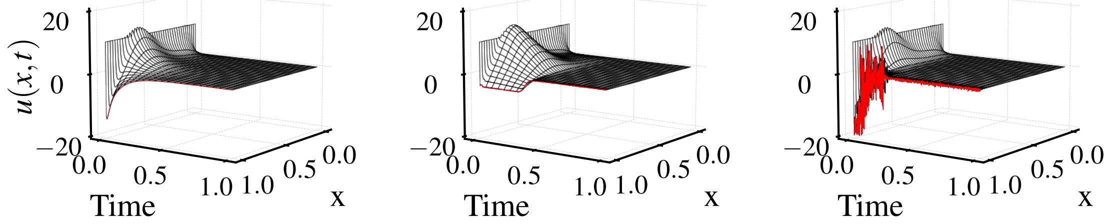

Figure 3: Example of reaction-diffusion PDE system stabilization using backstepping, PPO, and SAC (left to right) under initial conditions $u(x, 0) = 10$. The recirculation coefficient using the Chebyshev polynomial defined as $\lambda(x) = 50 \cos(\gamma \cos^{-1}(x))$ with $\gamma = 8$.

Experimental design We adopt the same approach as the 1D Hyperbolic PDE in Section 5.1 except that the dynamics are now governed by (8), (9), with full state measurements and $u(1, t) = U(t)$. The time-horizon is shortened to 1 second as the algorithms are able to stabilize faster. We choose $\lambda(x) = 50 \cos(\gamma \cos^{-1}(x))$ where γ is fixed to be 8 (future studies may vary γ). At each episode, the initial conditions are uniformly randomized according to $u(x, 0) \sim$ Uniform(1, 10), and we note that the system is always open-loop unstable for all possible initial conditions (See Figure 8 in Appendix). For training, we follow the same procedure as Section 5.1 except that we require a finer simulation resolution of $\Delta x = 0.005$, and the PDE is simulated at $\Delta t = 0.00001$ due to the approximation of the second spatial derivative in the reaction-diffusion PDE.

Results Figure 1 presents the average reward over 1 million training steps. Unlike the hyperbolic PDE where both RL algorithms performed relatively equal, the PPO algorithm achieved better performance during training which is corroborated by the testing rewards in the middle column of Table 3. Figure 3 demonstrates a test case with $u(x, 0) = 10$. Similar to the transport PDE, we observe oscillations in the RL feedback laws, suggesting potential improvement via enforcing continuity constraints as in Asadi et al. (2018). Notably, in Figure 3, perhaps due to reward shaping, PPO differs from the backstepping controller's approach, but maintains excellent performance.

5.3. 2D Navier-Stokes PDEs

Experimental design For the Navier-Stokes 2D problem (Section 4.3), both velocity components are zero initially, i.e., $u(x, y, 0) = v(x, y, 0) = 0$. We apply boundary control on the top boundary with tangential, uniform controlled velocity, setting $u(x, 1, t) = U(t) \in \mathbb{R}$ and $v(x, 1, t) = 0$. For implementation, we discretize the state space with a spatial step of $\Delta x = 0.05$ and the PDE is simulated at $\Delta t = 0.001$. The reward for training is derived from the negative of the cost in optimization (15). The reference velocity vector $\boldsymbol{u}_{ref}$ is the resulted velocity vector under the boundary control $U(t) = 3 - 5t$, and $U_{ref} = 2.0$.

Results Figure 1 (right) shows the average reward per episode for PPO and SAC, with PPO outperforming SAC both in terms of higher final rewards and more stable training curves. Table 3 (right) details the average episodic rewards over 50 test episodes, where PPO surpasses both SAC and the model-based optimization algorithm, which often gets stuck in local optima. Despite the reward difference, on a singular example presented in Figure 4, all methods effectively track the reference velocity vectors.

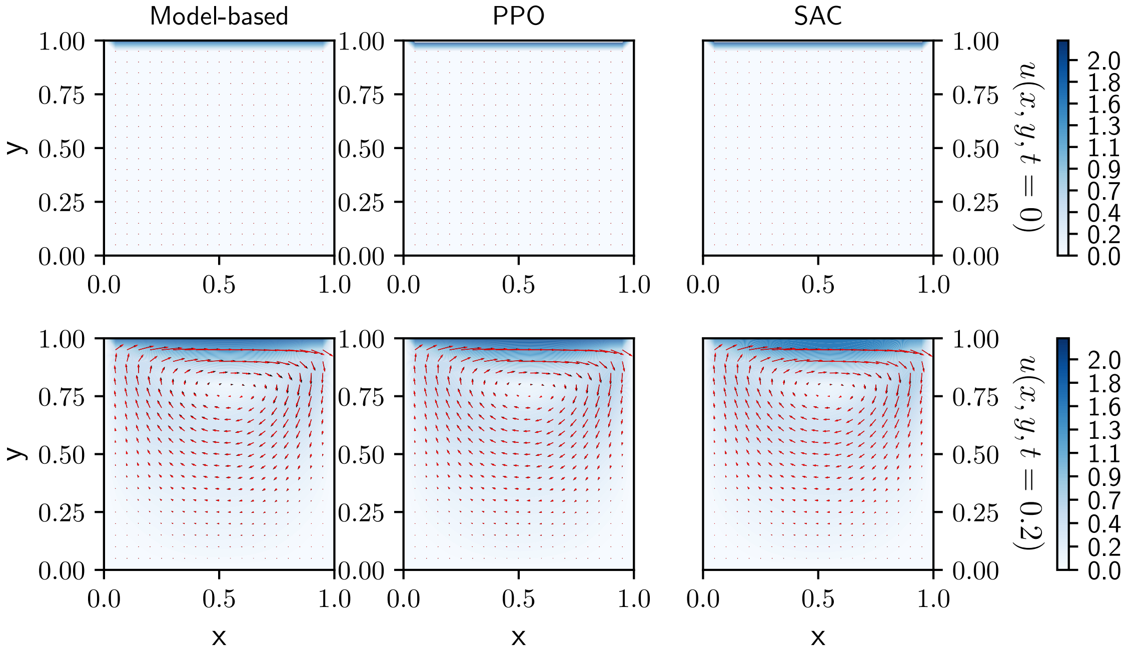

Figure 4: Example of Navier-Stokes PDE tracking using optimization-based control, PPO, and SAC under initial conditions $u(x, y, 0) = 0$ at $t = 0$ (top) and $t = 0.2$ (bottom). Red and black arrows represent the actual and reference velocity field respectively. The background color represents the magnitude of the velocity vector.

6. Conclusion

Future work Throughout this paper, we have mentioned several avenues for future research based on the PDE Control Gym. As such, we conclude this work by briefly summarizing these ideas. We employed relatively simple policy network architectures in our RL algorithms, not fully fine-tuning them to the specific problems. The PDE Control Gym presents opportunities to optimize policy network structures, improve reward shaping, and develop better RL algorithms for PDE control tasks. Additionally, our experiments were based on time-invariant linear instability coefficients where $\beta(x)$ and $\lambda(x)$ are unknown but static during RL training. Thus, there is much to be explored for model-free controllers when considering time-varying models, adaptive control, and sensing noise. Furthermore, given the superior performance of backstepping controllers, investigating the potential of pre-training RL methods through imitation learning could be a valuable direction.

Conclusion In this study, we introduced the first benchmark suite for learning-based boundary control of PDEs. We developed RL gyms for three fundamental PDE control problems: the 1D transport PDE, 1D reaction-diffusion PDE, and 2D Navier Stokes PDE. This gym allows for the separation of algorithm design from the numerical implementation of PDEs. Moreover, we trained a series of *model-free* RL models on the three benchmarks and compared their performance with model-based PDE backstepping and optimization methods. Finally, our work discussed multiple avenues for future research, aiming to inspire new research in the challenging field of PDE control.

References

Kavosh Asadi, Dipendra Misra, and Michael L. Littman. Lipschitz continuity in model-based reinforcement learning, 2018.

Julian Berberich, Carsten W. Scherer, and Frank Allgöwer. Combining prior knowledge and data for robust controller design. *IEEE Transactions on Automatic Control*, 68(8):4618–4633, 2023. doi: 10.1109/TAC.2022.3209342.

Luke Bhan, Yuanyuan Shi, and Miroslav Krstic. Neural operators for bypassing gain and control computations in PDE backstepping. *IEEE Transactions on Automatic Control*, pages 1–16, 2023a. doi: 10.1109/TAC.2023.3347499.

Luke Bhan, Yuanyuan Shi, and Miroslav Krstic. Operator learning for nonlinear adaptive control. In *Proceedings of The 5th Annual Learning for Dynamics and Control Conference (L4DC)*, volume 211 of *Proceedings of Machine Learning Research*, pages 346–357. PMLR, 15–16 Jun 2023b.

Yuexin Bian, Xiaohan Fu, Rajesh K Gupta, and Yuanyuan Shi. Ventilation and temperature control for energy-efficient and healthy buildings: A differentiable pde approach. *arXiv preprint arXiv:2403.08996*, 2024.

Lukas Brunke, Melissa Greeff, Adam W Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P Schoellig. Safe learning in robotics: From learning-based control to safe reinforcement learning. *Annual Review of Control, Robotics, and Autonomous Systems*, 5:411–444, 2022.

Xin Chen, Guannan Qu, Yujie Tang, Steven Low, and Na Li. Reinforcement learning for selective key applications in power systems: Recent advances and future challenges. *IEEE Transactions on Smart Grid*, 13(4):2935–2958, 2022.

Cenk Demir, Shumon Koga, and Miroslav Krstic. Neuron growth control and estimation by pde backstepping. *Automatica*, 165:111669, 2024. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2024.111669. URL https://www.sciencedirect.com/science/article/pii/S0005109824001626.

Florent Di Meglio, Rafael Vazquez, Miroslav Krstic, and Nicolas Petit. Backstepping stabilization of an underactuated 3 × 3 linear hyperbolic system of fluid flow equations. In *2012 American Control Conference (ACC)*, pages 3365–3370, 2012.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

Jie Feng, Yuanyuan Shi, Guannan Qu, Steven H. Low, Anima Anandkumar, and Adam Wierman. Stability constrained reinforcement learning for decentralized real-time voltage control. *IEEE Transactions on Control of Network Systems*, pages 1–12, 2023. doi: 10.1109/TCNS.2023.3338240.

Max D Gunzburger. *Perspectives in flow control and optimization*. SIAM, 2002.

Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized PDE modeling. arXiv:2209.15616, 2022.

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, 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018b. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

Philipp Holl, Nils Thuerey, and Vladlen Koltun. Learning to control PDEs with differentiable physics. In *International Conference on Learning Representations (ICLR)*, 2020.

Miroslav Krstic Huan Yu. *Traffic Congestion Control by PDE Backstepping*. Birkhäuser Cham, 2023.

Mojtaba Izadi, Javad Abdollahi, and Stevan S. Dubljevic. PDE backstepping control of one-dimensional heat equation with time-varying domain. *Automatica*, 54:41–48, 2015.

A. Jameson, L. Martinelli, and N. A. Pierce. Optimum aerodynamic design using the Navier-Stokes equations. *Theoretical and Computational Fluid Dynamics*, 10(1):213–237, Jan 1998.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2021.

Shumon Koga and Miroslav Krstic. *Materials Phase Change PDE Control & Estimation*. Birkhäuser, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, 2012.

Miroslav Krstic and Andrey Smyshlyaev. 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.

Miroslav Krstic and Andrey Smyshlyaev. *Boundary Control of PDEs*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2008b.

Miroslav Krstic, Luke Bhan, and Yuanyuan Shi. Neural operators of backstepping controller and observer gain functions for reaction–diffusion PDEs. *Automatica*, 164:111649, 2024. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2024.111649. URL https://www.sciencedirect.com/science/article/pii/S0005109824001420.

Hung Le and Parviz Moin. An improvement of fractional step methods for the incompressible Navier-Stokes equations. *Journal of computational physics*, 92(2):369–379, 1991.

Randall J. LeVeque. *Numerical methods for conservation laws (2. ed.)*. Lectures in mathematics. Birkhäuser, 1992. ISBN 978-3-7643-2723-1.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)*, 23(3):449–456, 2004.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Scott J Moura, Nalin A Chaturvedi, and Miroslav Krstić. Adaptive partial differential equation observer for battery state-of-charge/state-of-health estimation via an electrochemical model. *Journal of Dynamic Systems, Measurement, and Control*, 136(1):011015, 2014.

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

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.

Jie Qi, Jing Zhang, and Miroslav Krstic. Neural operators for delay-compensating control of hyperbolic PIDEs. arXiv:2307.11436, 2023.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

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

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

Yuanyuan Shi, Zongyi Li, Huan Yu, Drew Steeves, Anima Anandkumar, and Miroslav Krstic. Machine learning accelerated PDE backstepping observers. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 5423–5428, 2022.

A. Smyshlyaev and M. Krstic. 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.

A. Smyshlyaev and M. Krstic. *Adaptive Control of Parabolic PDEs*. Princeton University Press, 2010.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: an extensive benchmark for scientific machine learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 1596–1611, 2022.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind control suite. *CoRR*, abs/1801.00690, 2018.

Rafael Vazquez, Guangwei Chen, Junfei Qiao, and Miroslav Krstic. The power series method to compute backstepping kernel gains: Theory and practice. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pages 8162–8169, 2023. doi: 10.1109/CDC49753.2023.10384080.

Haiyan Wang, Feng Wang, and Kuai Xu. *Modeling information diffusion in online social networks with partial differential equations*, volume 7. Springer Nature, 2020.

Huan Yu, Saehong Park, Alexandre Bayen, Scott Moura, and Miroslav Krstic. Reinforcement learning versus PDE backstepping and PI control for congested freeway traffic. *IEEE Transactions on Control Systems Technology*, 30(4):1595–1611, 2022.

Xiangyuan Zhang, Weichao Mao, Saviz Mowlavi, Mouhacine Benosman, and Tamer Başar. Controlgym: Large-scale control environments for benchmarking reinforcement learning algorithms, 2024.

Appendix A. Hyperbolic Partial Differential Equation

A.1. First-Order Finite Difference Scheme

We consider the benchmark transport PDE in the form

$$u_t \quad = \quad u_x + \beta(x)u(0,t), \quad (x,t) \in [0,1) \times \mathbb{R}_+ \,, \tag{16}$$

with a single boundary condition representing the control input in either Dirichlet Neumann form at $x = 1$ $(u(1,t) = U(t))$. Notice that instability is caused by the recirculation term $\beta(x)u(0,t)$ - otherwise the resulting PDE with be considered an instantiation of the inviscid Burger's equation. For the numerical scheme, consider the first-order Taylor approximation for u and the resulting substitution of derivatives yields

$$u_j^{n+1} = u_j^n + \Delta t \left(\frac{u_{j+1}^n - u_j^n}{\Delta x} + \beta_j u_0^n \right) \,, \tag{17}$$

where Δt denotes the temporal timestep, Δx denotes the spatial timestep, $n = 0, ..., Nt$, $j = 0, ..., Nx$ where Nt and Nx are the total number of temporal and spatial steps respectively. To enforce Neumann boundary conditions, let $u_\zeta^n|_{\zeta=Nx}$ represent the spatial derivative at time t of spatial point Nx which is given by the user as control input. Then, we have

$$u_\zeta^n|_{\zeta=Nx} = \frac{u_{Nx}^n - u_{Nx-1}^n}{\Delta x} \,, \tag{18}$$

which is rearranged for the final boundary point

$$u_{Nx}^n = u_{Nx-1}^n + (\Delta x)u_\zeta^n|_{\zeta=Nx} \,. \tag{19}$$

In the case of Dirichlet boundary conditions, the computation is straightforward as u_{Nx}^n is directly set as the given control input. For stabilization of the finite-difference scheme, it is recommended that timestep be much much smaller than the spatial step LeVeque (1992).

A.2. Details on Numerical Implementations for Benchmark 1D Hyperoblic PDE

A.2.1. REINFORCEMENT LEARNING BASELINES: HYPERPARAMETERS FOR PROXIMAL POLICY OPTIMIZATION (PPO) AND SOFT-ACTOR CRITIC (SAC)

We provide the entire set of hyperparameters for training the PPO and SAC algorithms in Table 4 and 5 respectively. For a details on the RL learning algorithms, see Schulman et al. (2017) (PPO) and Haarnoja et al. (2018b) (SAC).

PPO Parameter	Value
Learning Rate	0.0003
Num Steps per Update	2048
Batch_size	64
Num Epcohs per Surrogate Loss Update	10
Discount Factor γ	0.99
Bias vs Variance Trade-Off for Advantage Estimator	0.95
Clipping Parameter ϵ	0.2
Entropy Coefficient for Loss	0.0
Value Function Coefficient for Loss	0.5
Max Value for Gradient Clipping	0.5

Table 4: Parameters for PPO Model Trained

SAC Parameter	Value
Learning Rate	0.0003
Buffer Size	1000000
Batch Size	256
Soft Update Coefficient τ	0.005
Discount factor γ	0.99
Action Noise	None

Table 5: Parameters for SAC Model Trained

A.2.2. EXPERIMENTAL DESIGN FOR 1D HYPERBOLIC PDE

We now discuss the construction of each episode for reinforcement learning which is concurrently summarized as a Markov Decision Process (MDP) in Table 6. For this work, we consider a simplified version of the PDE where γ is fixed for the entire training process to $\gamma = 7.35$. Then, each episode randomly begins with an initial condition sampled from $u(x, 0) \sim \text{Uniform}(1, 10)$. This creates a sufficiently challenging PDE to control as for any initial condition, the PDE is open loop unstable. As an example, we present the PDEs for $u(x, 0) = 1, 10$ in the left and right of Figure 5.

Markov Decision Process Tuple	Parameter for Hyperbolic PDE Benchmark Example
$\mathcal{S}$ - state space	$\mathcal{S} \subseteq C[0,1]$
$\mathcal{A}$ - action space	$\mathcal{A} \subseteq \{x \in \mathbb{R} : -20 \leq x \leq 20\}$
$\mathcal{O}$ - observation space	$\mathcal{O} = \mathcal{S}$ (can be varied to user choice)
p_0 - initial sample pdf for $u(x,0) \in \mathcal{S}$	$p_0 \subseteq \{f \in C[0,1] \mid f(x) = c, \quad \forall x \in [0,1]\}$ where $c \sim$ Uniform$(1,10)$
$p_f(\cdot \mid x_t, a_t)$ - state transition	Dynamics described by PDE in (5) with $u(1,t) = a_t$
T - time horizon	5 seconds
$r_a(s,s')$ - reward ($s \in \mathcal{S}$)	$r_a(s,s') = -1 * \lVert s' - s \rVert_{L_2}$
$q(s_T, a_0, ..., a_T)$ terminal cost ($s_T \in \mathcal{S}$)	$q(s_T, a_0, ..., a_T) = \begin{cases} 0 & \lVert s_T \rVert_{L_2} > \zeta \\ \sigma - 1/\eta * \sum_{\tau=0}^{T} \lvert a_\tau \rvert_{L_1} - \lVert s_t \rVert_{L_2} & \lVert s_T \rVert_{L_2} \leq \zeta \end{cases}$ where σ, η, ζ are hyperparameters

Table 6: Markov Decision Process describing the PDE example in Section A.2

For the RL policies, we utilized a novel reward function designed to accurately handle the challenging oscillations of hyperbolic PDEs (See Figure 6 with the backstepping controller for an example). With this, we developed the following reward function for 1D PDEs:

$$r_a(s,s') = -1 * \|s' - s\|_{L_2}, \tag{20}$$

$$q(s_T, a_0, ..., a_T) = \begin{cases} 0 & \|s_T\|_{L_2} > 20 \\ \sigma - 1/\eta * \sum_{\tau=0}^{T} |a_\tau|_{L_1} - \|s_t\|_{L_2} & \|s_T\|_{L_2} \leq 20 \end{cases}, \tag{21}$$

where σ, η, ζ were hyperparameters set to $300, 1000, 20$. The intuition of this reward function is in two components. First, $r_a(s,s')$ rewards control inputs that force the state to become smaller compared to its previous state leading to an encouragement towards stabilization. Then, at the termination of the episode, if the final state L_2 norm is small enough (≤ 20), we give the policy a large reward which is now penalized by the control costs - ie, we only value control costs for the policy after we know that the policy for the episode is stabilizing. This trade-off is commonly found in a series of optimal control problems - but due to the challenging nature of PDEs, one is usually satisfied with just stabilization guarantees without limits on the control inputs. This highlights one of the advantages of using model-free approaches where one can begin to enforce a stabilization first, minimization of control second approach. Lastly, we note that many rewards will encourage successful policies and we leave it to the community to develop better tuned rewards for these benchmark PDE problems.

For training, we run each model with parameters listed in Tables 4 and 5 (standard defaults for stable-baselines3, non-optimized) in the environment governed by the MDP in Table 6. From an implementation perspective, we discretize both the state space using a spatial step of $dx = 0.01$ and further discretize each timestep of the MDP by $dt = 0.01$ as well. However, we note that the first-order PDE scheme in Section A.1 requires a much smaller timestep then spatial step and thus, the PDE is internally simulated at a timestep of $dt_{PDE} = 0.0001$ where it receives a new control input at every 0.01 second and maintains the same input for the higher resolution PDE simulation until the next 0.01 second is reached. Naturally, it is worth considering sampling of the controller at the same rate of the PDE - but this results in poor performance for the reinforcement learning models and is less indicative of a real-world application given it is challenging to provide control inputs at frequencies larger than 100Hz.

A.3. Case Study: Hyperbolic PDE

In this section, we show the results for two individual initial conditions for the Hyperbolic PDE problem, namely $u(x, 0) = 0$ and $u(x, 0) = 10$.

Initial Condition	Control Algorithm	Total Episode Reward ↑	Total Episode State Summed L_2 Norm $(\sum_{t=0}^{T} \|u(\cdot, t)\|_{L_2}) \downarrow$
$u(x, 0) = 1$	Backstepping	**289.8**	**106.1**
$u(x, 0) = 1$	PPO	246.0	448.1
$u(x, 0) = 1$	SAC	212.9	720.4
$u(x, 0) = 10$	Backstepping	**198.4**	**1060.9**
$u(x, 0) = 10$	PPO	32.7	2026.4
$u(x, 0) = 10$	SAC	133.7	1402.8

Table 7: Resulting control performance for backstepping, PPO, and SAC on the two PDE examples corresponding to Figure 6.

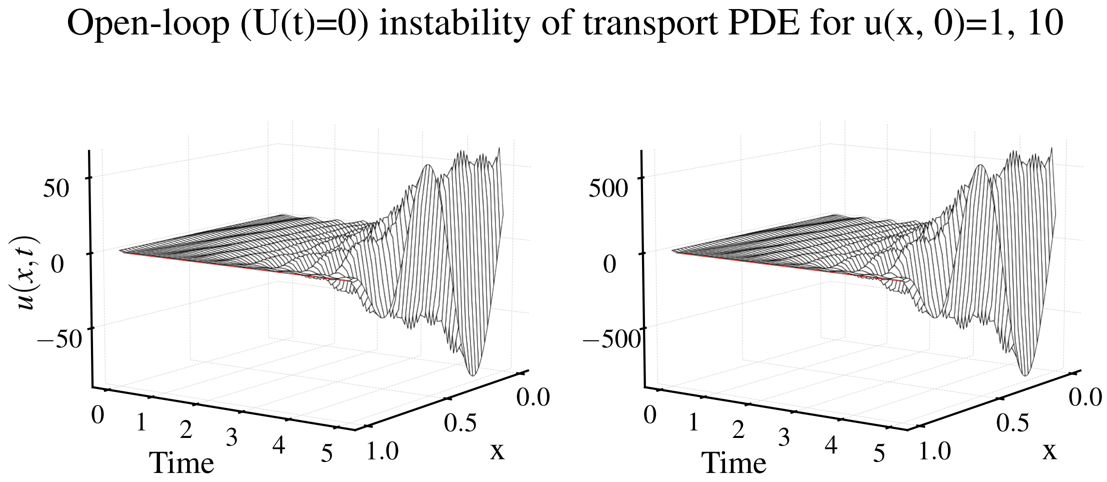

Figure 5: Instability of the 1D transport PDE with $\beta(x) = 5\cos(7.35\cos^{-1}(x))$ under a openloop control signal ($U(t) = 0$).

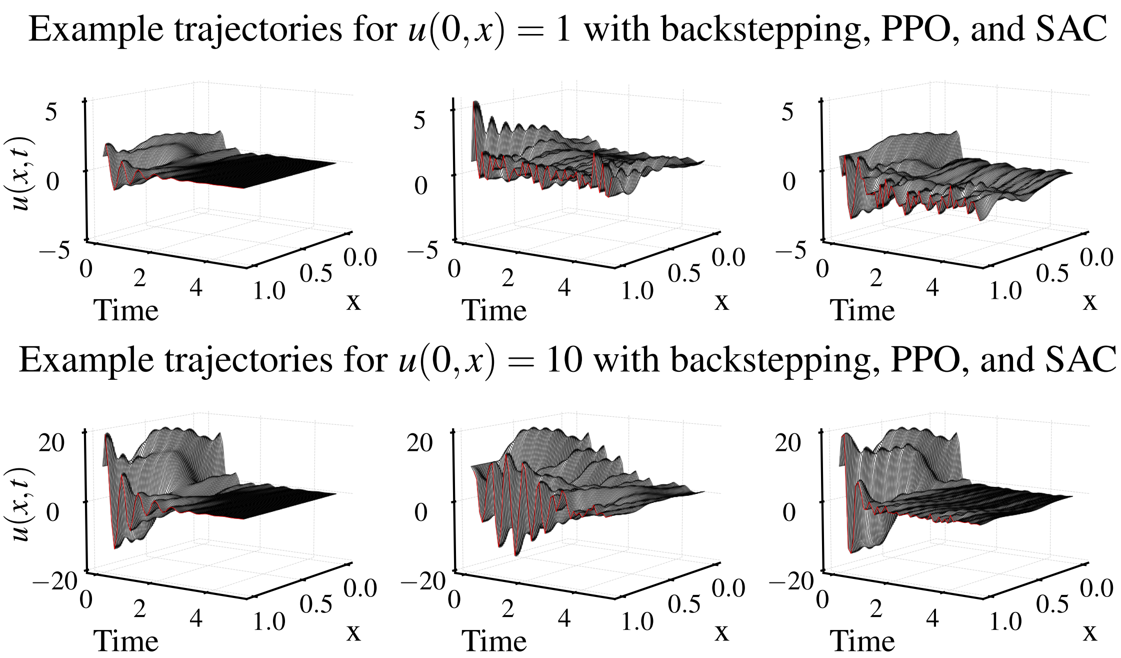

Figure 6: Examples of PDE system stabilization using backstepping, PPO, and SAC (left to right) under two different initial conditions $u(x,0) = 1, 10$. The PDE has functional recirculation coefficient using the Chebyhsev polynomial defined as $\beta(x) = 5\cos(\gamma \cos^{-1}(x))$ with $\gamma = 7.35$. The control signal for each approach is marked in red and in Figure 7.

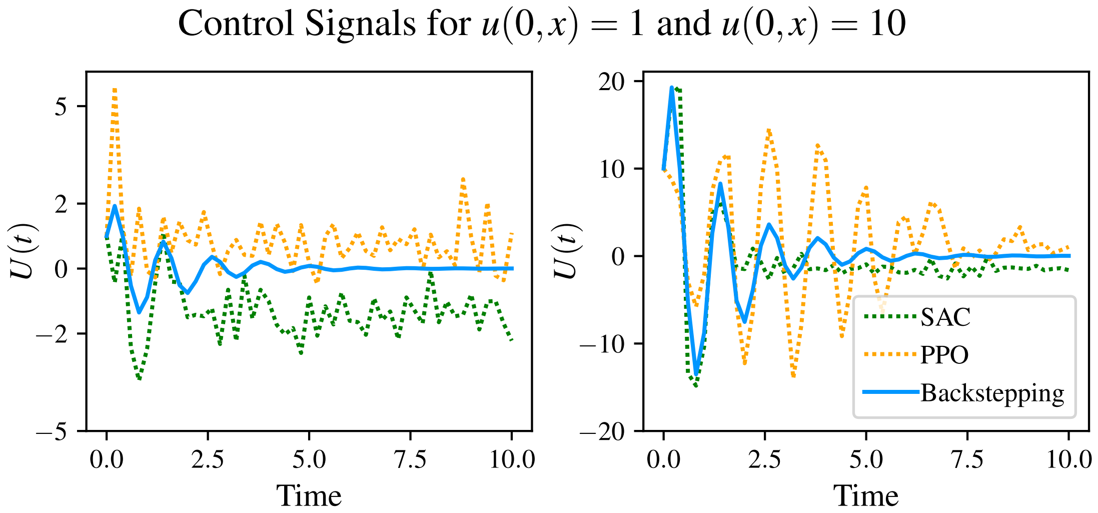

Figure 7: The corresponding control signal given to the PDEs in Figure 6. The control signals on the left match the PDE with $u(x,0) = 1$ (top-row of Fig. 6) and the signals on the right correspond to the PDE with $u(x,0) = 10$ (bottom-row of Fig. 6) respectively.

Appendix B. Reaction-Diffusion Partial Differential Equation

B.1. First-Order Finite Difference Scheme

We consider the benchmark reaction-diffusion PDE in the form

$$\begin{aligned} u_t &= u_{xx} + \lambda(x) u(x,t), \quad (x,t) \in (0,1) \times \mathbb{R}_+ \,, && (22) \\ u(0,t) &= 0 \,, && (23) \end{aligned}$$

with the control input again in either Dirichlet or Neumann form at $x = 1$ ($u(1,t) = U(t)$). As with the transport PDE, the recirculation term $\lambda(x)u(x,t)$ causes instability as otherwise the resulting PDE would reduce to the stable heat equation. For the numerical scheme, consider the first-order Taylor approximation for u as

$$u_j^{n+1} = u_j^n + \Delta t \left(\frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{(\Delta x)^2} + \lambda_j u_j^n \right) , \tag{24}$$

where Δt denotes the temporal timestep, Δx denotes the spatial timestep, $n = 0, ..., Nt$, $j = 0, ..., Nx$ where Nt and Nx are the total number of time steps respectively. Explicitly we set $u_0^k = 0 \forall k \in [0, Nt]$ for the first boundary condition and the second boundary condition follows as in the Hyperbolic PDE case with (19) for Neumann boundary conditions and u_{Nx}^n for Dirichlet boundary conditions. Again, we require extremely small spatial and temporal time-steps for stabilization of the scheme.

B.2. Details on numerical implementation for benchmark 1D reaction-diffusion PDE

We adopt the exact same, **untuned**, hyperparameters for PPO and SAC as in Section A.2.1 and adopt the same MDP as presented in Table 6 except that the dynamics are now governed by (22), (23), with $u(1,t) = a_t$ and the time-horizon is shortened to 1 second as the algorithms are able to stabilize faster. We choose $\lambda(x) = 50\cos(\gamma \cos^{-1}(x))$ where γ is fixed to be 8. With the following configuration and initial conditions again sampled between 1 and 10, we see that the PDE is openloop unstable in Figure 8.

For training, we follow the same procedure as Section A.2 except that we require a finer simulation resolution of $dt = 0.001$, $dx = 0.005$, and $dt_{PDE} = 0.00001$ due to the approximation of the second spatial derivative in the reaction diffusion PDE.

B.2.1. CASE STUDY: PARABOLIC PDE

In this section, we show the results for two individual initial conditions for the parabolic PDE problem, namely $u(x, 0) = 0$ and $u(x, 0) = 10$.

Initial Condition	**Control Algorithm**	**Total Episode Reward** ↑	**Total Episode Summed L_2 Norm** ($\sum_{t=0}^{T} \|u(\cdot, t)\|_{L_2}$) ↓
$u(x, 0) = 1$	Backstepping	**299.8**	1275.4
$u(x, 0) = 1$	PPO	295.1	**1103.5**
$u(x, 0) = 1$	SAC	239.8	1968.7
$u(x, 0) = 10$	Backstepping	**298.2**	12754.4
$u(x, 0) = 10$	PPO	283.2	23342.4
$u(x, 0) = 10$	SAC	140.5	**9624.1**

Table 8: Resulting control performance for backstepping, PPO, and SAC on the two PDE examples corresponding to Figure 9.

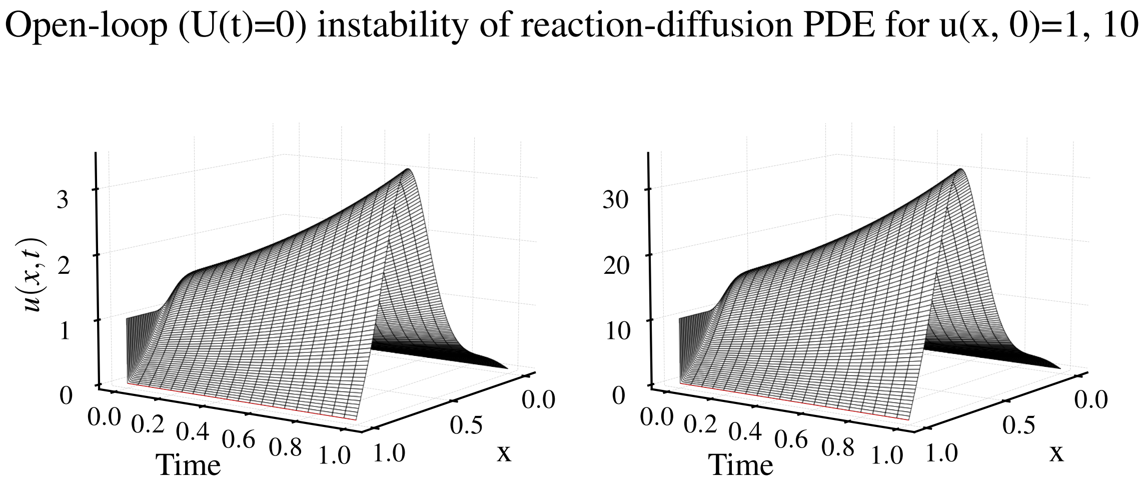
Open-loop (U(t)=0) instability of reaction-diffusion PDE for u(x, 0)=1, 10

Figure 8: Instability of the reaction-diffusion with $\beta(x) = 50 \cos(8 \cos^{-1}(x))$ under a openloop control signal ($U(t) = 0$).

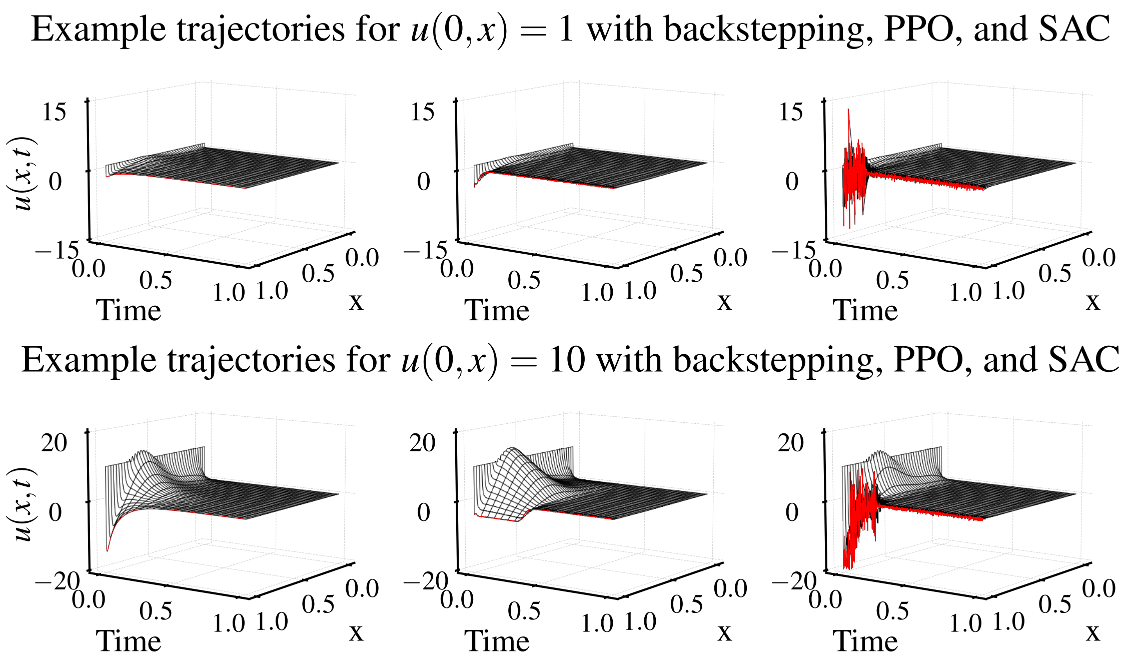

Figure 9: Examples of PDE system stabilization using backstepping, PPO, and SAC (left to right) under two different initial conditions $u(x, 0) = 1, 10$. The PDE has functional recirculation coefficient using the Chebyhsev polynomial defined as $\beta(x) = 50 \cos(\gamma \cos^{-1}(x))$ with $\gamma = 8$. The control signal for each approach is marked in red and in Figure 10.

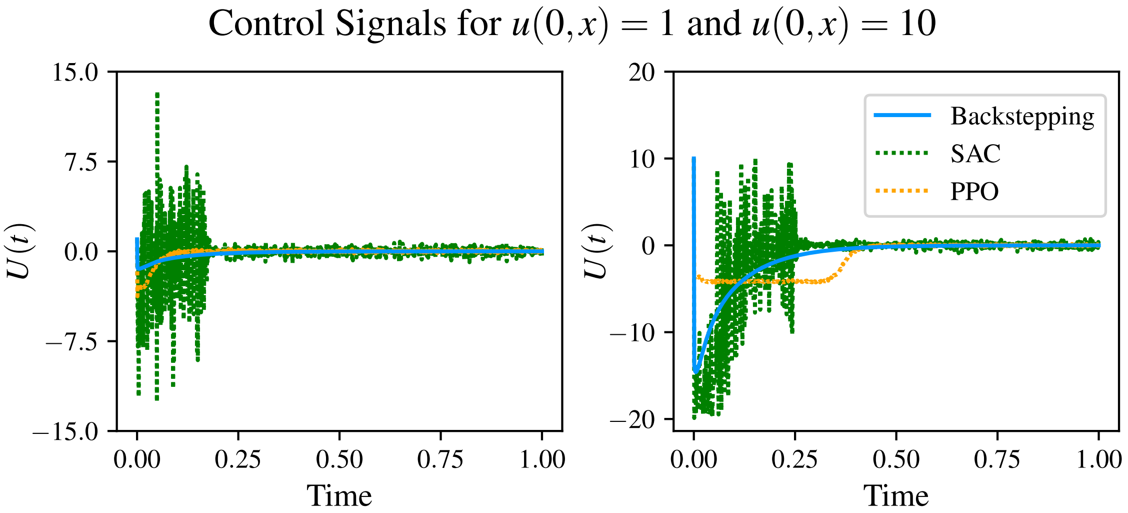

Figure 10: The corresponding control signal given to the PDEs in Figure 9. The control signals on the left match the PDE with $u(x, 0) = 1$ (top-row of Fig. 9) and the signals on the right correspond to the PDE with $u(x, 0) = 10$ (bottom-row of Fig. 9) respectively.

Appendix C. Navier Stokes Equation

C.1. Second-Order Finite Difference Scheme

We consider 2D benchmark Navier Stokes PDE in the form:

$$\nabla \cdot \boldsymbol{u} = 0\,, \qquad (x,y,t) \in ([0,1]\times[0,1]\times\mathbb{R}_+)\,, \tag{25a}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\boldsymbol{u}\,, \qquad (x,y,t) \in ((0,1)\times(0,1)\times\mathbb{R}_+)\,, \tag{25b}$$

$$\boldsymbol{u}(x,0,t) = U(x,t)\,, \qquad (x,t) \in ([0,1]\times\mathbb{R}_+)\,, \tag{25c}$$

$$\boldsymbol{u}(x,1,t) = \boldsymbol{u}(1,y,t) = \boldsymbol{u}(0,y,t) = 0\,, \qquad (x,y,t) \in ([0,1]\times[0,1]\times\mathbb{R}_+)\,, \tag{25d}$$

with boundary control input in the top boundary and the velocity at all other boundaries is set to be zero. We incorporate a predictor-corrector Le and Moin (1991) scheme for the imcompressive Navier Stokes equations. In the scheme, we denote $\boldsymbol{u} = (u, v)$ is the velocity vector, and spatial position is $\boldsymbol{x} = (x, y)$.

We first present the predictor step that does not consider the pressure,

$$\begin{aligned} u^*_{i,j} = u^n_{i,j} &+ \Delta t(\nu(\frac{u^n_{i-1,j} - 2u^n_{i,j} + u^n_{i+1,j}}{(\Delta x)^2} + \frac{u^n_{i,j-1} - 2u^n_{i,j} + u^n_{i,j+1}}{(\Delta y)^2})) \\ &- \Delta t(u^n_{i,j}\frac{u^n_{i+1,j} - u^n_{i-1,j}}{2\Delta x} + v^n_{i,j}\frac{u^n_{i,j+1} - u^n_{i,j-1}}{2\Delta y})), \end{aligned} \tag{26}$$

$$\begin{aligned} v^*_{i,j} = v^n_{i,j} &+ \Delta t(\nu(\frac{v^n_{i-1,j} - 2v^n_{i,j} + v^n_{i+1,j}}{(\Delta x)^2} + \frac{v^n_{i,j-1} - 2v^n_{i,j} + v^n_{i,j+1}}{(\Delta y)^2})) \\ &- \Delta t(u^n_{i,j}\frac{v^n_{i+1,j} - v^n_{i-1,j}}{2\Delta x} + v^n_{i,j}\frac{v^n_{i,j+1} - v^n_{i,j-1}}{2\Delta y})\,, \end{aligned} \tag{27}$$

where Δt denotes the temporal timestep, $\Delta x, \Delta y$ denotes the spatial time step, $n = 0, \ldots Nt$, $i = 0, \ldots, Nx$, $j = 0, \ldots, Ny$ where Nt, Nx, Ny are the total number of timesteps, respectively. $u^*_{i,j}, v^*_{i,j}$ are the resulted velocity field that does consider pressure.

To satisfy continuity equation $\nabla \cdot \boldsymbol{u} = 0$, we get the pressure Poisson equation:

$$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = -\rho(\frac{\partial^2 u}{\partial x^2} + 2\frac{\partial u}{\partial x}\frac{\partial v}{\partial y} + \frac{\partial^2 v}{\partial y^2})\,. \tag{28}$$

Then, we iteratively solve the Poisson equation for pressure:

$$\begin{aligned} p^{iter+1}_{i,j} \leftarrow &\frac{1}{2(\Delta x^2 + \Delta y^2)}\left((p^{iter}_{i+1,j} + p^{iter}_{i-1,j})\Delta y^2 + (p^{iter}_{i,j+1} + p^{iter}_{i,j-1})\Delta x^2\right) \\ &+ \rho(\frac{u^*_{i+1,j} - u^*_{i-1,j}}{2\Delta x} + \frac{v^*_{i+1,j} - v^*_{i-1,j}}{2\Delta y})\Delta x^2\Delta y^2\,, \end{aligned} \tag{29}$$

with $iter$ denotes the iteration of the procedure for solving pressure. We denote the pressure solution as p^*. Then we can perform the corrector step to compute the velocoty vector at next time step:

$$u^{n+1}_{i,j} = u^*_{i,j} - \Delta t \cdot \tfrac{1}{\rho}\tfrac{p^*_{i+1,j} - p^*_{i-1,j}}{2\Delta x}\,, \tag{30}$$

$$v^{n+1}_{i,j} = v^*_{i,j} - \Delta t \cdot \tfrac{1}{\rho}\tfrac{p^*_{i,j+1} - p^*_{i,j-1}}{\Delta y}\,. \tag{31}$$

We apply boundary conditions every time step after the predictor step and corrector step.

C.2. Control Algorithms Implemented for Navier Stokes PDE

C.2.1. MODEL-BASED OPTIMIZATION

In optimization algorithms, the optimal control is computed as solution of an optimization problem, where the partial differential equations appear as equality constraints. In the experiment, we consider to track the velocity vector to be the desired trajectory $\boldsymbol{u}_{ref}(\boldsymbol{x}, t)$ and to minimize the control cost with a reference control U_{ref}. The optimization problem (i.e. optimal control) is formulated as follows,

$$\begin{aligned}
\min_{U(t)} \quad & J(U(t), \boldsymbol{u}), && (32a)\\
\text{s.t.} \quad & \frac{\partial \boldsymbol{u}}{\partial t} = -\boldsymbol{u} \cdot \nabla \boldsymbol{u} + -\frac{1}{\rho}\nabla p - \nu \nabla^2 \boldsymbol{u}, && (32b)\\
& \nabla \cdot \boldsymbol{u} = 0, && (32c)\\
& \boldsymbol{u}(\boldsymbol{x} \in \Gamma, t) = U(t) && (32d)
\end{aligned}$$

where Γ is the defined controllable boundary. The optimization problem can be solved using the augmented Lagrangian:

$$\begin{aligned}
L(U, \boldsymbol{u}, \boldsymbol{\lambda}(\boldsymbol{x}, t), \nu(\boldsymbol{x}, t)) \quad = \quad & J(U(t), \boldsymbol{u}) + \iint\limits_A \mu(\nabla \cdot \boldsymbol{u}) \\
& + \iint\limits_A \boldsymbol{\lambda}^T (\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u}). \qquad (33)
\end{aligned}$$

A locally optimal solution is characterized by satisfying the first-order Karush-Kuhn-Tucker (KKT) conditions, which necessitate that the derivatives of the Lagrangian function L with respect to variables $U, \boldsymbol{u}, \boldsymbol{\lambda}, \mu$ are all zero. Specifically, taking the derivatives with respect to $\boldsymbol{\lambda}$ and μ yields the Navier-Stokes Equations. Additionally, the solution $\boldsymbol{\lambda}$ is required to be divergence-free, a condition that arises from the derivative of L with respect to the pressure p. The derivative of L with respect to $\boldsymbol{u}$ leads to the differential equation for $\boldsymbol{\lambda}$:

$$\frac{\partial}{\partial t}\boldsymbol{\lambda} = -(G + G^T)\boldsymbol{u} - \nu \nabla^2 \boldsymbol{\lambda} - \nabla \mu + (\boldsymbol{u} - \boldsymbol{u}_{ref}), \qquad (34)$$

where $G = \frac{\partial \boldsymbol{\lambda}}{\partial \boldsymbol{x}}$ represents the Jacobian of $\boldsymbol{\lambda}$ with respect to $\boldsymbol{x}$. The differential equation can be solve backwards in time with $\boldsymbol{\lambda}(\boldsymbol{x}, T) = 0$ and homogeneous boundary conditions Pyta et al. (2015). The derivative with respect to U leads to

$$U = U_{ref} - \frac{\nu}{\gamma} \oint_\Gamma \frac{\partial \boldsymbol{\lambda}_1}{\partial \boldsymbol{x}_2}. \qquad (35)$$

Thus the (local) optimal control function $U(t)$ can be computed by 1) Solving Navier Stokes equation to get the velocity vector $\boldsymbol{u}(x, y, t)$ 2) Solving adjoint equation (34) for $\boldsymbol{\lambda}(x, y, t)$ backwards from time T to 0 with $\boldsymbol{u}(x, y, t)$ 3) Solving equation (35) for $U(t)$ forwards with $\boldsymbol{\lambda}(x, y, t)$.

C.2.2. REINFORCEMENT LEARNING BASELINES: PROXIMAL POLICY OPTIMIZATION (PPO) AND SOFT-ACTOR CRITIC (SAC)

We adopt the exact same, **untuned**, hyperparameters for PPO and SAC as in Section A.2.1.

C.3. Details on numerical implementation for benchmark 2D Navier Stokes PDE

We adopt the MDP as presented in Table 9. For training, we run each model with parameters listed in 4, 5 (standard defaults for stable-baselines3, non-optimized) in the environment governed by the MDP in Table 9. From an implementation perspective, we discretize both the state space using a spatial step of $dx = 0.05$ and further discretize each timestep of the MDP by $dt = 0.001$ as well. The PDE is internally simulated at a timestep of $dt_{PDE} = dt = 0.001$. We use the negative of the cost to be the reward for training. The reference velocity vector s_{ref} is the resulted velocity vector under the boundary control $U(t) = 3 - 5t$, and the reference control $u_{ref} = a_{ref} = 2$.

Markov Decision Process Tuple	Parameter for Navier Stokes PDE Benchmark Example
$\mathcal{S}$ - state space	$\mathcal{S} \subseteq C([0,1] \times [0,1])$
$\mathcal{A}$ - action space	$\mathcal{A} \subseteq \{x \in \mathbb{R} : -10 \leq x \leq 10\}$
$\mathcal{O}$ - observation space	$\mathcal{O} = \mathcal{S}$ (can be modified to partially observable)
p_0 - initial sample	$p_0 \subseteq \{f \in C[0,1] \times [0,1] \| f(x,y) = (0,0) \, \forall x \in [0,1], y \in [0,1]\}$
$p_f(\cdot \| x_t, a_t)$ - state transition	Dynamics described by PDE in (25) with boundary control $u(x,0,t) = a_t$ and $v(x,0,t) = 0$
T - time horizon	0.2 seconds
$r_a(s, s')$ - reward ($s \in \mathcal{S}$)	$r_a(s,s') = -\frac{1}{2} * \|s' - s_{ref}\|_{L_2} - \gamma \frac{1}{2} \|a - a_{ref}\|_{L_2}, \gamma = 0.1, a_{ref} = 2$

Table 9: Markov Decision Process describing the PDE example in Section C.3