CL-DIFFPHYCON: CLOSED-LOOP DIFFUSION CONTROL OF COMPLEX PHYSICAL SYSTEMS

Long Wei[1*] **Haodong Feng**[1*] **Yuchen Yang**[2§] **Ruiqi Feng**[1] **Peiyan Hu**[3§]
Xiang Zheng[4§] **Tao Zhang**[1] **Dixia Fan**[1] **Tailin Wu**[1†]
[1]Department of Artificial Intelligence, Westlake University,
[2]School of Statistics and Data Science, Nankai University,
[3]Academy of Mathematics and Systems Science, Chinese Academy of Sciences,
[4]School of Future Technology, South China University of Technology
{weilong,fenghaodong,wutailin}@westlake.edu.cn

ABSTRACT

The control problems of complex physical systems have broad applications in science and engineering. Previous studies have shown that generative control methods based on diffusion models offer significant advantages for solving these problems. However, existing generative control approaches face challenges in both performance and efficiency when extended to the closed-loop setting, which is essential for effective control. In this paper, we propose an efficient Closed-Loop Diffusion method for Physical systems Control (CL-DiffPhyCon). By employing an asynchronous denoising framework for different physical time steps, CL-DiffPhyCon generates control signals conditioned on real-time feedback from the system with significantly reduced computational cost during sampling. Additionally, the control process could be further accelerated by incorporating fast sampling techniques, such as DDIM. We evaluate CL-DiffPhyCon on two tasks: 1D Burgers' equation control and 2D incompressible fluid control. The results demonstrate that CL-DiffPhyCon achieves superior control performance with significant improvements in sampling efficiency.

1 INTRODUCTION

The control problem of complex physical systems is a critical area of study that involves optimizing a sequence of control actions to achieve specific objectives. It has important applications across a wide range of science and engineering fields, including fluid control (Verma et al., 2018), plasma control (Degrave et al., 2022), and particle dynamics control (Reyes Garza et al., 2023). The challenge in controlling such systems arises from their high-dimensional, highly nonlinear, and stochastic characteristics. Therefore, to achieve good control performance, there is an inherent requirement of *closed-loop* control. Specifically, each control decision should be based on the latest state feedback determined by the system dynamics, allowing for continuous adaptation of the control inputs in response to any changes. This requirement is particularly necessary for control tasks that involve extra challenges, such as partial observations or indirect control.

Over recent decades, several methods have been developed to address this problem, including classical control methods, recent reinforcement learning approaches, and the latest generative methods. Among them, diffusion models, such as DiffPhyCon (Wei et al., 2024), have demonstrated competitive performance, often outperforming both classical control and reinforcement learning methods in complex physical systems control. The superiority of diffusion models for general decision-making problems has also been widely demonstrated recently (Ajay et al., 2022; Janner et al., 2022a).

However, these diffusion control approaches encounter significant challenges in handling the closed-loop control problems, due to their reliance on a synchronous denoising strategy. The diffusion models start from pure noise to a denoised sample for all physical time steps within the model horizon. Applying a full sampling process at each physical time step can realize the closed-loop control, but

[*]Equal contribution. [§]Work done as an intern at Westlake University. [†]Corresponding author.

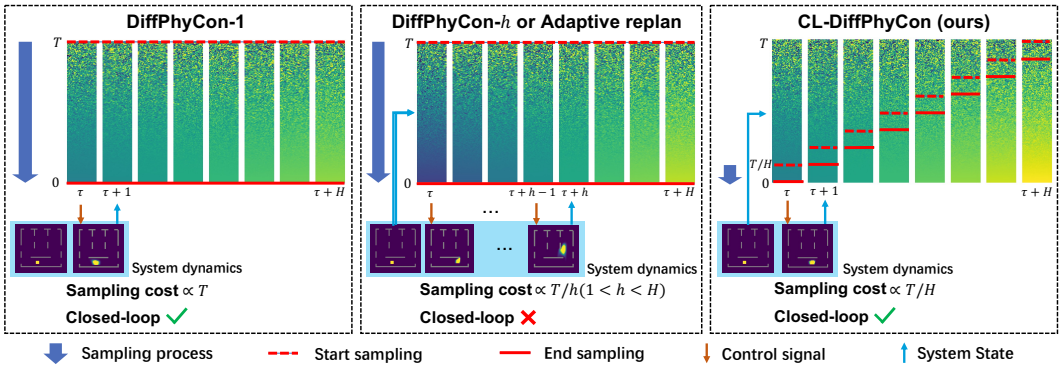

Figure 1: **Advantages of our CL-DiffPhyCon (right) over previous diffusion control methods (left and middle).** By employing an asynchronous denoising framework, our method could achieve closed-loop control and accelerate the sampling process significantly. The notation DiffPhyCon-h means conducting a full sampling process including T denoising steps every h physical time steps.

incurs high sampling cost. Moreover, it may disrupt the consistency of the control signals, thereby affecting overall performance (Kaelbling & Lozano-Pérez, 2011). On the other hand, conducting a full sampling process every several physical time steps improves sampling efficiency but no longer conforms to the closed-loop requirement, leading to inferior control decisions due to outdated system states. Although an online replanning strategy has been proposed recently to determine when to replan adaptively (Zhou et al., 2024), they do not establish a fully closed-loop framework. In addition, it involves extra computation of likelihood estimation or sampling from scratch, with a dependence on thresholding hyperparameters, which may vary across different tasks and require experiments or a significant amount of tuning to determine.

In this paper, we propose a novel Closed-Loop Diffusion method for Physical systems Control, named as CL-DiffPhyCon. The key idea is to decouple the synchronous denoising within the model horizon, allowing different physical time steps to exhibit different noise levels. In this way, closed-loop generation of control sequences is naturally realized: the asynchronous diffusion model outputs control signals sequentially with increasing levels of noise along physical time steps, which enables utilization of real-time feedback state for control signal sampling in each horizon without waiting for all the following control signals in the same horizon to be denoised completely. Then, the feedback serves as the initial condition for sampling subsequent control signals, ensuring they are generated based on this reliable state. Our method can also be seen as a seamless replanning approach that leverages fresh observations with minimal sampling costs. Therefore, compared to existing diffusion-based control methods (Wei et al., 2024; Zhou et al., 2024), our approach not only realizes closed-loop control but also achieves significant sampling acceleration. These advantages of our method are illustrated in Figure 1.

In summary, we make the following contributions: (1) We propose CL-DiffPhyCon, a novel closed-loop diffusion control method for complex physical systems. The core of this method is an asynchronous diffusion model derived from a theoretical analysis of the target distribution. This model enables parallel denoising across physical time steps, allowing earlier control actions to be sampled sooner, thereby accelerating the sampling process. Additionally, the control process could be further accelerated by incorporating fast sampling techniques, such as DDIM (Song et al., 2020). (2) We evaluate CL-DiffPhyCon on the 1D Burgers' equation control and 2D incompressible fluid control tasks. The results demonstrate that CL-DiffPhyCon achieves notable control performance with significant sampling acceleration.

2 Related Work

Classical control methods like PID (Li et al., 2006) and MPC (Schwenzer et al., 2021) are known for their high efficiency, steady performance, and good interpretability, but they face significant challenges in both performance and efficiency when applied to control high-dimensional complex physical systems. Recently, imitation learning and reinforcement learning has shown good perfor-

mance on a wide range of physical systems (Pomerleau, 1988; Zhuang et al., 2023), including drag reduction (Rabault et al., 2019; Elhawary, 2020; Feng et al., 2023; Wang et al., 2024), heat transfer (Beintema et al., 2020; Hachem et al., 2021), and fish swimming (Novati et al., 2017; Verma et al., 2018; Feng et al., 2024). Another category of supervised learning (SL) methods (Holl et al., 2020; Hwang et al., 2022) plan control signals by utilizing backpropagation through a neural surrogate model. In contrast, our approach does not depend on surrogate models; instead, it simultaneously learns the dynamics of physical systems and the corresponding control sequences. Additionally, physics-informed neural networks (PINNs) (Willard et al., 2020) have recently been utilized for control (Mowlavi & Nabi, 2023), but they necessitate explicit PDEs. In contrast, our method is data-driven and has broader applicability.

Diffusion models (Ho et al., 2020) excel at learning high-dimensional distributions and have achieved significant success in image and video generation (Dhariwal & Nichol, 2021; Ho et al., 2022), weather forecasting (Price et al., 2023), and inverse design (Wu et al., 2024), to name a few. They have also demonstrated superior capabilities on decision making tasks, such as robot control (Janner et al., 2022b; Ajay et al., 2022) and high-dimensional nonlinear systems control (Wei et al., 2024), compared with widely used reinforcement learning and imitation learning approaches. However, they struggle to balance the conflicting goals of achieving closed-loop control and maintaining efficient sampling for long trajectories. Some previous work has focused on improving the adaptability of diffusion generation (Zhou et al., 2024) on decision-making tasks, but it is not closed-loop and needs extra hyperparameters and computations for the decision of replanning. In contrast, our method is a closed-loop approach with an efficient sampling strategy without extra hyperparameter. Some recent works also assign varying noise levels to different frames within a model horizon (Wu et al., 2023; Ruhe et al., 2024). The key differences between our work and these approaches are twofold: (1) Our method focuses on the closed-loop control task of complex physical systems, whereas their methods are geared towards sequential content generation that does not involve interaction with the external world. (2) The diffusion models we need to learn are derived from the target distribution we aim to sample from (see Section 4.1 for details), while theirs are heuristic.

3 Background

3.1 Problem Setup

Given an initial state $\mathbf{u}_0$, a system dynamics G, and a specified control objective $\mathcal{J}$, We consider the following complex physical systems control problem:

$$\min_{\pi} \mathbb{E}_{\mathbf{w}_{\tau+1}\sim\pi(\mathbf{w}_{\tau+1}|\mathbf{u}_\tau)}[\mathcal{J}(\mathbf{u}_0, \mathbf{w}_1, \mathbf{u}_1, \ldots, \mathbf{w}_N, \mathbf{u}_N)] \quad \text{s.t.} \quad \mathbf{u}_{\tau+1} = G(\mathbf{u}_\tau, \mathbf{w}_{\tau+1}, \xi_\tau). \tag{1}$$

Here $\mathbf{u}_\tau \in \mathbb{R}^{d_\mathbf{u}}$ and $\mathbf{w}_\tau \in \mathbb{R}^{d_\mathbf{w}}$ are the **system state** and external **control signal** at physical time step τ, respectively. The **system dynamics** G represents the transition of states over time under external control in the system, typically determined by implicit PDEs. G could be *stochastic* with nonzero random noise ξ_τ, or *deterministic* with $\xi_\tau = 0$. The evolution of states can only be observed through state measurement. The **control objective** $\mathcal{J}$ is defined over a trajectory of length N, representing the performance of the control strategy. For example, $\mathcal{J}$ can be designed to measure the deviation from a target state $\mathbf{u}^*$, subject to cost constraints: $\mathcal{J} = \|\mathbf{u}_N - \mathbf{u}^*\|^2 + \sum_{\tau=1}^{N} \|\mathbf{w}_\tau\|^2$. In this paper, we focus on *closed-loop* control, which means that the control signal $\mathbf{w}_{\tau+1}$ in each time step is sampled from a distribution conditioned on the current state $\mathbf{u}_\tau$. Unlike open-loop control, which determines all actions in advance, closed-loop control continuously incorporates real-time feedback to adjust control decisions dynamically, making it particularly effective for complex and evolving systems where evolution of states can only be observed through measurement of the state. To simplify notation, we introduce a variable $\mathbf{z}_\tau = [\mathbf{w}_\tau, \mathbf{u}_\tau]$ to represent the concatenation of $\mathbf{w}_\tau$ and $\mathbf{u}_\tau$. The transition probabilities in the training trajectories collected offline are assumed to satisfy the *Markov property*:

$$p(\mathbf{z}_{\tau+1}|\mathbf{u}_0, \mathbf{z}_{1:\tau}) = p(\mathbf{z}_{\tau+1}|\mathbf{u}_\tau) = p(\mathbf{w}_{\tau+1}|\mathbf{u}_\tau)p(\mathbf{u}_{\tau+1}|\mathbf{u}_\tau, \mathbf{w}_{\tau+1}). \tag{2}$$

3.2 Preliminary: Diffusion Control Models

DiffPhyCon (Wei et al., 2024) is a recent diffusion generative method to solve the problem Eq. 1 for small N in the open-loop manner. In this section, we briefly review this framework of its

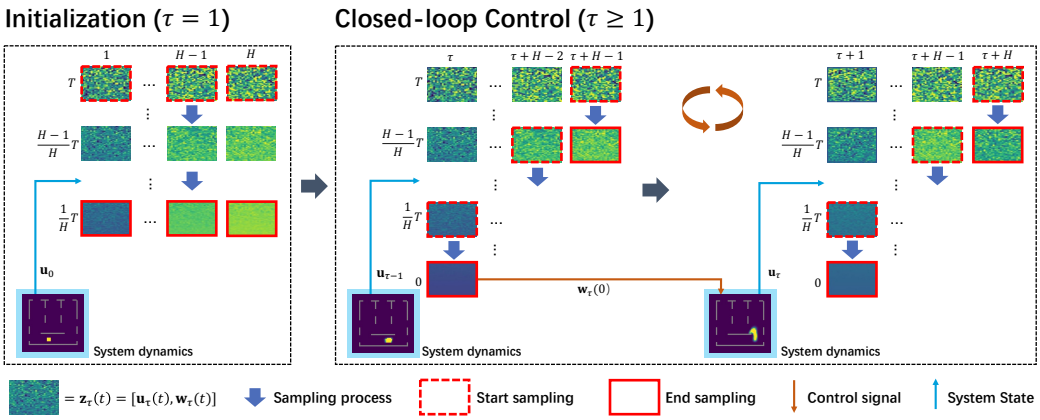

Figure 2: **CL-DiffPhyCon for closed-loop control**. First, it uses the synchronous diffusion model for initialization. Then, it uses the asynchronous diffusion model for iterative control. Sampling of each control signal is based on the latest state feedback from the system dynamics.

light version. Suppose that we have a training set $\mathcal{D}_{\text{train}}$ containing M trajectories $\{\mathbf{u}_0^{(i)}, \mathbf{z}_{1:N}^{(i)}\}_{i=1}^M$ collected offline. We define the following forward diffusion SDE (Song et al., 2021) on the $\mathcal{D}_{\text{train}}$,

$$\mathrm{d}\mathbf{z}_\tau(t) = f(t)\mathbf{z}_\tau(t)\mathrm{d}t + g(t)\mathrm{d}\omega_\tau(t), \quad \tau \in [1:N], \quad t \in [0, T], \tag{3}$$

where $\mathbf{z}_\tau(0) = \mathbf{z}_\tau$, $f(t)$ and $g(t)$ are scalar functions, and $\omega_\tau(t)$ is Wiener process [1]. Through Eq. 3, we augment the distribution of $\mathcal{D}_{\text{train}}$, denoted as $p_{\text{train}}(\mathbf{u}_0, \mathbf{z}_{1:N}) = p_{\text{train}}(\mathbf{u}_0)p_{\text{train}}(\mathbf{z}_{1:N}|\mathbf{u}_0)$, to the distribution of the diffusion process $p_t(\mathbf{u}_0, \mathbf{z}_{1:N}(t)) = p_{\text{train}}(\mathbf{u}_0)p_t(\mathbf{z}_{1:N}(t)|\mathbf{u}_0)$. We have the terminal conditions, $p_0(\mathbf{z}_{1:N}(0)|\mathbf{u}_0) = p_{\text{train}}(\mathbf{z}_{1:N}(0)|\mathbf{u}_0)$ and $p_T(\mathbf{z}_{1:N}(T)|\mathbf{u}_0) \approx \mathcal{N}(\mathbf{z}_{1:N}(T)|\mathbf{0}, \sigma_T^2\mathbf{I})$. The reverse-time SDE of Eq. 3 has the following formula:

$$\mathrm{d}\mathbf{z}_{1:N}(t) = [f(t)\mathbf{z}_{1:N}(t) - g(t)^2\nabla \log p_t(\mathbf{z}_{1:N}(t)|\mathbf{u}_0)]\mathrm{d}t + g(t)\mathrm{d}\omega_{1:N}(t), \quad t \in [T, 0]. \tag{4}$$

Once we have the score function $\nabla \log p_t(\mathbf{z}_{1:N}(t)|\mathbf{u}_0)$, we can sample $\mathbf{z}_{1:N}(0) \sim p_{\text{train}}(\mathbf{z}_{1:N}(0)|\mathbf{u}_0)$ from $\mathbf{z}_{1:N}(T) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ through Eq. 4. According to DiffPhyCon, the control objective $\mathcal{J}(\mathbf{z}_{1:N}(0))$ is achieved by solving the inverse problem (Song et al., 2021; Chung et al., 2023):

$$\begin{aligned}\mathrm{d}\mathbf{z}_{1:N}(t) =& [f(t)\mathbf{z}_{1:N}(t) - g(t)^2\nabla_{\mathbf{z}_{1:N}(t)} \log p_t(\mathbf{z}_{1:N}(t)|\mathbf{u}_0) \\ &+ g(t)^2\lambda \cdot \nabla_{\mathbf{z}_{1:N}(t)}\mathcal{J}(\hat{\mathbf{z}}_{1:N}(0))]\mathrm{d}t + g(t)\mathrm{d}\omega_{1:N}(t).\end{aligned} \tag{5}$$

Here, $\hat{\mathbf{z}}_{1:N}(0)$ is Tweedie's estimate, $s(t)^{-1}\mathbf{z}_{1:N}(t) + s(t)\sigma(t)^2\nabla_{\mathbf{z}_{1:N}(t)} \log p_t(\mathbf{z}_{1:N}(t)|\mathbf{u}_0)$, where $s(t) = \exp\left(\int_0^t f(\eta)\mathrm{d}\eta\right)$ and $\sigma(t) = \sqrt{\int_0^t \frac{g(\eta)^2}{s(\eta)^2}\mathrm{d}\eta}$. In practical implementation, these quantities are specialized by VP SDE (Song et al., 2021), i.e. DDPM (Ho et al., 2020), and the diffusion model is implemented by a parameterized denoising network $\boldsymbol{\epsilon}_\phi$ of horizon H, which equals N. Please refer to Appendices A and B for more details.

4 METHOD

In this section, we detail our method CL-DiffPhyCon. In Section 4.1, we illustrate our idea and derive the two distributions we need to learn. To sample from them, we present the synchronous and asynchronous diffusion model in Section 4.2 and Section 4.3, respectively. In Section 4.4, we introduce closed-loop control (inference) by CL-DiffPhyCon, which is also illustrated in Figure 2.

4.1 ASYNCHRONOUSLY DENOISING FRAMEWORK

Recall that DiffPhyCon (Wei et al., 2024) requires latent variables, e.g., system state and control signal, during the reverse diffusion process of horizon H (which is typically much shorter than N)

[1] In $\mathbf{z}_\tau(t)$, the subscript τ denotes physical time step, and (t) in the parentheses indicates SDE step.

being denoised synchronously. As a result, to sample a control signal, a full denoising process of length T over the whole horizon is performed, introducing a large amount of computation over all the latent variables within this horizon. To address this issue, we propose CL-DiffPhyCon, an *asynchronous denoising process* scheme such that the latent variables of the early physical time step are denoised in advance of the latter ones. At each physical time step, the sampled control signal is input to the system dynamics and the output state serves as the initial condition for the following denoising process. Thus, the control signal is planned based on the current system state and closed-loop control is achieved. Meanwhile, the computational cost between two successive times steps is reduced significantly compared with synchronous sampling.

Formally, we aim to model the joint distribution $p\big(\mathbf{z}_1(0), \mathbf{z}_2(0), \cdots, \mathbf{z}_N(0)|\mathbf{u}_0\big)$ in a *Markov* complex physical system, where each $\mathbf{z}_\tau(0) = [\mathbf{w}_\tau, \mathbf{u}_\tau]$ is a pair of noise-free control signal and system state, and $\mathbf{u}_0$ is the initial state. Denote $\mathbf{z}_{\tau:\tau+H-1}(t) = [\mathbf{z}_\tau(t), \mathbf{z}_{\tau+1}(t), \cdots, \mathbf{z}_{\tau+H-1}(t)]$ as the sequence of hidden variables with the same noise level in the horizontal interval $[\tau : \tau + H - 1]$ and $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t) = [\mathbf{z}_\tau(t), \mathbf{z}_{\tau+1}(t + \frac{1}{H}T), \cdots, \mathbf{z}_{\tau+H-1}(t + \frac{H-1}{H}T)]$ as its counterpart with *asynchronous* noise level. Let's consider the augmented joint distribution

$$p\big(\mathbf{z}_{1:N}(0), \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \cdots, \tilde{\mathbf{z}}_{N+1:N+H}(\frac{1}{H}T)|\mathbf{u}_0\big). \tag{6}$$

Through sampling from this augmented joint distribution, we can obtain the desired control signals and states sequence $\mathbf{z}_{1:N}(0) \sim p_{\text{train}}\big(\mathbf{z}_{1:N}(0)|\mathbf{u}_0\big)$. By conditioning on previous variables sequentially, we obtain the following decomposition theorem:

Theorem 1. *Assume that the joint distribution* $p\big(\mathbf{z}_1(0), \mathbf{z}_2(0), \cdots, \mathbf{z}_N(0)|\mathbf{u}_0\big)$ *has* Markov property. *For any* $\tau > 0$*, we assume* $\mathbf{z}_\tau(T)$ *is independently normally distributed with density* $\mathcal{N}(\mathbf{z}_\tau(T)|\mathbf{0}, \sigma_T^2\mathbf{I})$*. The augmented joint distribution can be decomposed as:*

$$\begin{aligned} &p\big(\mathbf{z}_{1:N}(0), \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \cdots, \tilde{\mathbf{z}}_{N+1:N+H}(\frac{1}{H}T)|\mathbf{u}_0\big) \\ =&p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big) \prod_{\tau=1}^{N} p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\mathcal{N}(\mathbf{z}_{\tau+H}(T); \mathbf{0}, \sigma_T^2\mathbf{I}). \end{aligned} \tag{7}$$

The proofs of this theorem and the subsequent propositions are provided in the Appendix C. This theorem implies that to sample from the augmented joint distribution, we only need to specify two kinds of distributions for ancestral sampling: $p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big)$, and $p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)$ for $\tau > 0$. Namely, we need one diffusion model to learn the initializing distribution $p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big)$, and another one to learn the transition distribution $p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)$.

4.2 Sampling from initializing distribution

To sample from $p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big)$, we train a *synchronous diffusion model* $\boldsymbol{\epsilon}_\phi$ by the following loss:

$$\mathcal{L}_{\text{synch}} = \mathbb{E}_{t,(\mathbf{u}_0,\mathbf{z}_{1:H}),\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi\big(\mathbf{z}_{1:H}(t), \mathbf{u}_0, t\big)\|_2^2]. \tag{8}$$

Here, $\mathbf{z}_{1:H}(t) = s(t)\mathbf{z}_{1:H} + s(t)^2\sigma(t)^2\boldsymbol{\epsilon}$, and the expectation is about $t \sim U(0, T)$, $(\mathbf{u}_0, \mathbf{z}_{1:H}) \sim \mathcal{D}_{\text{train}}$ and $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Since the negative score function minimizes this loss, we get $\nabla_{\mathbf{z}_{1:H}(t)} \log p_t\big(\mathbf{z}_{1:H}(t)|\mathbf{u}_0\big) \approx -\boldsymbol{\epsilon}_{\phi^*}\big(\mathbf{z}_{1:H}(t), \mathbf{u}_0, t\big)$ with the trained parameters ϕ^*. After training, we can apply Eq. 4 to sample $\{\mathbf{z}_{1:H}(t)\}$, where t goes from T to $\frac{1}{H}T$, from which we get $\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T) = [\mathbf{z}_1(\frac{1}{H}T), \cdots, \mathbf{z}_H(T)]$ following the desired distribution $p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big)$.

4.3 Sampling from transition distribution

To sample from $p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)$, we train an *asynchronous diffusion model* $\boldsymbol{\epsilon}_\theta$. Using the notations in Section 3.2, we define an asynchronous forward diffusion SDE:

$$\begin{aligned} \mathrm{d}\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t) =&[\mathrm{d}\mathbf{z}_\tau(t), \mathrm{d}\mathbf{z}_{\tau+1}(t + \frac{1}{H}T), \cdots, \mathrm{d}\mathbf{z}_{\tau+H-1}(t + \frac{H-1}{H}T)] \\ =&\tilde{f}_{\tau:\tau+H-1}(t)\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)\mathrm{d}t + \tilde{g}_{\tau:\tau+H-1}(t)\mathrm{d}\omega_{\tau:\tau+H-1}(t), \end{aligned} \tag{9}$$

where $\tilde{f}_{\tau:\tau+H-1}(t) = [f(t), f(t+\frac{1}{H}T), \cdots, f(t+\frac{H-1}{H}T)]$ and $\tilde{g}_{\tau:\tau+H-1}(t) = [g(t), g(t+\frac{1}{H}T), \cdots, g(t+\frac{H-1}{H}T)]$. The reverse-time SDE of Eq. 9 has the following formula:

$$\begin{aligned} \mathrm{d}\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t) =& [\tilde{f}_{\tau:\tau+H-1}(t)\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t) \\ & -\tilde{g}_{\tau:\tau+H-1}(t)^2 \nabla_{\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)} \log p_t\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)\big)]\mathrm{d}t \\ & +\tilde{g}_{\tau:\tau+H-1}(t)\mathrm{d}\omega_{\tau:\tau+H-1}(t). \end{aligned} \tag{10}$$

Further, the inference involving the control objective can be generalized as follows:

$$\begin{aligned} \mathrm{d}\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t) =& [\tilde{f}_{\tau:\tau+H-1}(t)\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t) \\ & -\tilde{g}_{\tau:\tau+H-1}(t)^2 \nabla_{\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)} \log p_t\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)\big) \\ & +\tilde{g}_{\tau:\tau+H-1}(t)^2 \lambda \nabla_{\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)} \mathcal{J}(\hat{\mathbf{z}}_{\tau:\tau+H-1}(0))]\mathrm{d}t \\ & +\tilde{g}_{\tau:\tau+H-1}(t)\mathrm{d}\omega_{\tau:\tau+H-1}(t). \end{aligned} \tag{11}$$

Here, $\hat{\mathbf{z}}_{\tau:\tau+H-1}(0)$ is an asynchronous Tweedie's estimate (detailed in Appendix A):

$$\hat{\mathbf{z}}_{\tau+i}(0) = s(t+\frac{i}{H}T)^{-1}\mathbf{z}_{\tau+i}(t+\frac{i}{H}T) + s(t+\frac{i}{H}T)\sigma(t+\frac{i}{H}T)^2 \frac{\partial \log p_t\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)\big)}{\partial \mathbf{z}_{\tau+i}(t+\frac{i}{H}T)}.$$

The distribution of $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)$ satisfies the following proposition:

Proposition 1. *Assume that the joint distribution* $p\big(\mathbf{z}_1(0), \mathbf{z}_2(0), \cdots, \mathbf{z}_N(0)|\mathbf{u}_0\big)$ *has* Markov property. *For any* $t \in [0, \frac{1}{H}T]$*, we have:*

$$p_t\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)\big) = \mathbb{E}_{\mathbf{z}_{\tau:\tau+H-1}(0)}[\prod_{i=0}^{H-1} p\big(\mathbf{z}_{\tau+i}(t+\frac{i}{H}T)|\mathbf{z}_{\tau+i}(0)\big)], \tag{12}$$

where the expectation is about $\mathbf{z}_{\tau:\tau+H-1}(0) \sim p_{\text{train}}\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)$*, and each*

$$p\big(\mathbf{z}_{\tau+i}(t+\frac{i}{H}T)|\mathbf{z}_{\tau+i}(0)\big) = \mathcal{N}\big(\mathbf{z}_{\tau+i}(t+\frac{i}{H}T); s(t+\frac{i}{H}T)\mathbf{z}_{\tau+i}(0), s(t+\frac{i}{H}T)^2\sigma(t+\frac{i}{H}T)^2\mathbf{I}\big).$$

Thus, we can obtain $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)$ for the following loss:

$$\mathcal{L}_{\text{asynch}} = \mathbb{E}_{\tau,t,(\mathbf{u}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)),\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t), \mathbf{u}_{\tau-1}, t\big)\|_2^2]. \tag{13}$$

Here, the expectation in Eq. 13 is about $\tau \sim U(1, N-H+1)$, $t \sim U(0, \frac{1}{H}T)$, $(\mathbf{u}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)) \sim p_{\text{train}}\big(\mathbf{u}_{\tau-1}(0)\big)p_t\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)\big)$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. After training, we can compute the score function by $\nabla_{\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)} \log p_t(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)) \approx -\boldsymbol{\epsilon}_{\theta^*}\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t), \mathbf{u}_{\tau-1}(0), t\big)$ with the trained parameters θ^*. Therefore, we can use Eq. 10, where t goes from $\frac{1}{H}T$ to 0, to sample $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0) \sim p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)$.

4.4 Closed-loop Control

During inference (control), our goal is not only generating control sequences and system trajectories that conform to physical dynamics but also optimizing our control objective. Based on the two learned models, we now introduce the closed-loop control procedure using $\boldsymbol{\epsilon}_\phi$ and $\boldsymbol{\epsilon}_\theta$, which together realize the distribution π in Eq. 1, under the guidance of $\mathcal{J}$. We first use the synchronous model $\boldsymbol{\epsilon}_\phi$ to produce the initial asynchronous variable $\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)$ conditioned on the initial state $\mathbf{u}_{\text{env},0}$ by applying Eq. 5 in the horizon $[1, H]$. Then, the control process starts. At each physical time step $\tau \geq 1$, we sample $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)$ using $\mathbf{u}_{\text{env},\tau-1}$ and $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)$ through Eq. 11. We extract the control signal $\mathbf{w}_\tau(0)$ from the sampled $\mathbf{z}_\tau(0)$, and input the pair $(\mathbf{u}_{\text{env},\tau-1}, \mathbf{w}_\tau(0))$ to the system dynamics G, which outputs the next state $\mathbf{u}_{\text{env},\tau}$[2]. Then we sample a noise $\mathbf{z}_{\tau+H}(T) \sim \mathcal{N}(\mathbf{0}, \sigma_T^2\mathbf{I})$, and concatenate it with the last $H-1$ components of $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)$ to compose $\tilde{\mathbf{z}}_{\tau+1:\tau+H}(\frac{1}{H}T)$. Now we take $\mathbf{u}_{\text{env},\tau-1}$, instead of the sampled $\mathbf{u}_{\tau-1}(0)$, and $\tilde{\mathbf{z}}_{\tau+1:\tau+H}(\frac{1}{H}T)$ to the next loop. The whole procedure is presented in Algorithm 1 and also illustrated in Figure 2. Consequently, the joint distribution of $\{\mathbf{w}_\tau(0), \mathbf{u}_{\text{env},\tau}\}_{\tau=1}^N$ satisfies the following proposition.

[2]In Section 4.4 and Algorithm 1, the subscript "env" in $\mathbf{u}_{\text{env},\tau}$ denotes the state feedback from the system, to distinguish $\mathbf{u}_{\text{env},\tau}$ from the sampled state $\mathbf{u}_\tau(0)$ by diffusion models.

Algorithm 1 Closed-loop Control of CL-DiffPhyCon

Require synchronous model $\boldsymbol{\epsilon}_\phi$, asynchronous model $\boldsymbol{\epsilon}_\theta$, control objective $\mathcal{J}(\cdot)$, initial state $\mathbf{u}_{\text{env},0}$, episode length N, model horizon H, full denoising steps T, hyperparameters λ.
Initialize $\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)$ using $\mathbf{u}_{\text{env},0}$, $\boldsymbol{\epsilon}_\phi$, and $\mathbf{z}_{1:H}(T) \sim \mathcal{N}(\mathbf{0}, \sigma_T^2\mathbf{I})$ by Eq. 5
for $\tau = 1, \cdots, N$ **do**
  for $t = T/H, T/H-1, \cdots, 1$ **do**
    update $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t-1)$ using $\mathbf{u}_{\text{env},\tau-1}(0)$, $\boldsymbol{\epsilon}_\theta$, and $\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)$ by Eq. 11
  end for
  $[\mathbf{u}_\tau(0), \mathbf{w}_\tau(0)] = \mathbf{z}_\tau(0)$
  input $\mathbf{u}_{\text{env},\tau-1}(0)$ and $\mathbf{w}_\tau(0)$ into the system dynamics G, which outputs $\mathbf{u}_{\text{env},\tau}$ // closed-loop feedback
  sample $\mathbf{z}_{t+H}(T) \sim \mathcal{N}(0, \sigma_T^2\mathbf{I})$ // append the end of the horizon with noise
  $\mathbf{z}_{\tau+1:\tau+H}(\frac{1}{H}T) = [\mathbf{z}_{\tau+1}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau+H}(T)]$
end for

Proposition 2. *Using inference described in Algorithm 1, the following holds:*

$$
\begin{aligned}
&p\big(\mathbf{w}_1(0), \mathbf{u}_{\text{env},1}, \cdots, \mathbf{w}_N(0), \mathbf{u}_{\text{env},N} | \mathbf{u}_{\text{env},0}\big) \\
=& \int p_{\text{gd}}\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T) | \mathbf{u}_{\text{env},0}\big) \\
&\prod_{\tau=1}^{N} p_{\text{gd}}\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0) | \mathbf{u}_{\text{env},\tau-1}, \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big) p_{\text{G}}\big(\mathbf{u}_{\text{env},\tau} | \mathbf{u}_{\text{env},\tau-1}, \mathbf{w}_\tau(0)\big) \\
&\mathcal{N}(\mathbf{z}_{\tau+H}(T); \mathbf{0}, \sigma_T^2\mathbf{I}) \mathrm{d}\{\mathbf{u}_0(0), \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \cdots, \mathbf{u}_N(0), \tilde{\mathbf{z}}_{N+1:N+H}(\frac{1}{H}T)\},
\end{aligned}
\tag{14}
$$

where p_{gd} denotes the transition distribution of the guided sampling (Eq. 11), and p_G denotes the transition distribution of the system dynamics G.

According to Eq. 14, for every τ, the control signal $\mathbf{w}_\tau(0)$ is conditional on the states $\mathbf{u}_{\text{env},0:\tau-1}$ instead of the predicted $\mathbf{u}_{0:\tau-1}(0)$. Therefore, our method achieves closed-loop control. It is clear that CL-DiffPhyCon is H/h times faster than DiffPhyCon-h, the control method that conducts a full sampling process of DiffPhyCon with horizon H every h physical time steps, as illustrated in Figure 1. Even compared with the adaptive replaning diffusion method (Zhou et al., 2024), CL-DiffPhyCon is still more efficient because each latent variable is sampled only once and it does not involve extra computation such as likelihood estimation.

Additionally, although some fast sampling methods were proposed recently, such as DDIM Song et al. (2020), to reduce sampling cost of diffusion models, our CL-DiffPhyCon still has an independent acceleration effect beyond them. The key insight is that CL-DiffPhyCon actually does not reduce the total number of sampling steps in each physical time step, which brings the opportunity to incorporate DDIM. Hence, by introducing DDIM in both the sampling processes of the synchronous and asynchronous models, the control efficiency of CL-DiffPhyCon could be further enhanced.

5 Experiment

In the experiments, we aim to answer three questions: (1) Can CL-DiffPhyCon outperform the classical and state-of-the-art baselines? (2) Can CL-DiffPhyCon achieve the desired acceleration of inference as we analyzed in Section 4.4, compared with diffusion control baselines, and obtain further acceleration by involving DDIM (Song et al., 2020)? (3) Can CL-DiffPhyCon address the challenges of noise, partial observation, partial/boundary control, and high dimensional indirect control? To answer these questions, we conduct experiments on two control tasks: 1D Burgers' equation control and 2D incompressible fluid control, both of which have important applications in science and engineering.

5.1 Baselines

The following classical, imitation learning, reinforcement learning, and diffusion control methods are selected as baselines: the classical control algorithm PID (Li et al., 2006); an imitation learning

Table 1: Comparison results on 1D Burgers' equation control. The average control objective ($\mathcal{J}$) and inference time (averaged across all settings) are reported, using a single NVIDIA A100 80GB GPU with 16 CPU cores. Bold font is the best model and the runner-up is underlined.

	noise-free ↓	physical constraint ↓	system noise ↓	measure noise ↓	FOPC ↓	POPC ↓	average time (s) ↓
BC	0.4708	0.4704	0.4138	0.3981	0.1093	0.0987	0.8356
BPPO	0.4686	0.4507	0.4088	0.3979	0.1079	0.0984	0.8231
PID	0.3250	0.3585	0.2323	0.2911	-	0.0827	**0.7717**
DiffPhyCon-1	0.0210	0.0214	0.0222	0.0232	0.0330	0.0332	49.2347
DiffPhyCon-5	0.0252	0.0257	0.0271	0.0272	0.0482	0.0484	10.8308
DiffPhyCon-15	0.0361	0.0365	0.0382	0.0377	0.1128	0.1132	4.9833
RDM	0.0296	0.0296	0.0310	0.0336	0.1124	0.1121	6.9130
CL-DiffPhyCon (ours)	**0.0096**	**0.0110**	**0.0095**	**0.0127**	**0.0291**	**0.0295**	4.5474
CL-DiffPhyCon (DDIM, ours)	0.0112	0.0123	0.0114	0.0146	0.0311	0.0313	0.8257

method Behaviour Cloning (BC) (Pomerleau, 1988); a recent reinforcement learning method Behavior Proximal Policy Optimization (BPPO) (Zhuang et al., 2023); two diffusion control methods, including RDM (Zhou et al., 2024), which adaptively decides when to conduct a full sampling process of control sequence, and DiffPhyCon (Wei et al., 2024), whose original version plans the control sequences of the whole trajectories in one sampling process. Since the trajectory length is much longer than the diffusion model horizon H ($H = 16$ and $H = 15$ in 1D and 2D tasks, respectively), we extend DiffPhyCon to its three variants DiffPhyCon-h ($h \in \{1, 5, H-1\}$) as our baselines, based on the predefined interval h of physical time steps to conduct a full sampling process. All the diffusion control baselines use the trained synchronous diffusion model of CL-DiffPhyCon for sampling, with steps $T = 900$ and $T = 600$ on the 1D and 2D tasks, respectively. PID is inapplicable to the complex 2D task (Åström & Hägglund, 2000). RDM is reproduced following the official code. However, the default values of two thresholding hyperparameters in RDM do not perform well on our tasks. Therefore, we select a pair of values that perform best (see Appendix I for details). For other baselines, we follow the implementations in DiffPhyCon Wei et al. (2024).

5.2 1D Burgers' Equation Control

Experiment settings. The Burgers' equation is a widely used equation to describe a variety of physical systems. We follow the works in Hwang et al. (2022); Mowlavi & Nabi (2023) and consider the 1D Burgers' equation with the Dirichlet boundary condition and external force $w(\tau, x)$ as follows:

$$\begin{cases} \frac{\partial u}{\partial \tau} = -u \cdot \frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2} + w(\tau, x) & \text{in } [0, \mathcal{T}] \times \Omega, \\ u(\tau, x) = 0 & \text{in } [0, \mathcal{T}] \times \partial\Omega, \\ u(0, x) = u_0(x) & \text{in } \{\tau = 0\} \times \Omega, \end{cases} \tag{15}$$

where ν is the viscosity, and $u_0(x)$ is the initial condition. Given a target state $u_d(\tau, x)$ defined in range $[0, \mathcal{T}] \times \Omega$, the control objective $\mathcal{J}$ is to minimize the error between $u(\tau, x)$ and $u_d(\tau, x)$:

$$\mathcal{J} \coloneqq \int_{\mathcal{T}} \int_{\Omega} |u(\tau, x) - u_d(\tau, x)|^2 \mathrm{d}x \mathrm{d}\tau, \tag{16}$$

subject to Eq. 15. We explore four kinds of settings from real-world considerations: (1) noise-free control; (2) control under physical constraint with a limited range of allowance for control actions; (3) control under random system and measurement noise, respectively; (4) partial control (PC) where actuators are limited to control approximately $1/8$ of the full spatial domain, which is further divided to full observation (FOPC) and partial observation (POPC) with sensors in $1/8$ of the full spatial domain. Details of these settings are provided in Appendix D.2.

Results. In Table 1, we present the results of our proposed CL-DiffPhyCon and baselines. Note that the reported metrics in different settings are not directly comparable. It can be seen that CL-DiffPhyCon delivers the best results compared to all baselines in all settings. BC and BPPO perform poorly on this task as they rely heavily on the quality of training control sequences, which are far from optimal solutions (see Appendix D.1). The diffusion control baselines perform better, compared with BC and BPPO, because diffusion models conduct global optimization over each horizon

Table 2: Comparison results on 2D incompressible fluid control. The average control objective ($\mathcal{J}$) and inference time (averaged across all settings) are reported, using a single NVIDIA A6000 48GB GPU with 16 CPU cores. Bold font is the best model and the runner-up is underlined.

	Large domain control		Boundary control		Average
	Fixed map↓	Random map↓	Fixed map↓	Random map↓	time (s) ↓
BC	0.6722	0.7046	0.8861	0.8871	4.67
BPPO	0.6343	0.6524	0.8830	0.8844	4.65
DiffPhyCon-1	0.5454	<u>0.3754</u>	0.7517	0.7955	1666.50
DiffPhyCon-5	0.5051	0.5458	0.6703	0.7451	357.66
DiffPhyCon-14	0.5823	0.5621	<u>0.6498</u>	0.7221	141.88
RDM	<u>0.4074</u>	0.4356	0.6553	<u>0.7087</u>	238.43
CL-DiffPhyCon (ours)	**0.3371**	**0.3485**	**0.6169**	**0.7003**	144.04
CL-DiffPhyCon (DDIM, ours)	0.4100	0.4254	0.6671	0.7109	26.01

through the conditional generation and excel in sampling from high dimensional space (Wei et al., 2024). Specifically, CL-DiffPhyCon decreases $\mathcal{J}$ by 54.3% and 48.6% compared with the best baselines (DiffPhyCon-1) in the noise-free and physical constraint settings, respectively. Furthermore, when the system or measurement is perturbed by noise, our method achieves improvements comparable to those in the noise-free case, outperforming the baselines by at least 48.6% and 57.2%. Moreover, despite the reduced controllable range in FOPC and POPC settings typically diminishes the effectiveness of diffusion-based control methods, CL-DiffPhyCon maintains superior performance over all baselines with at least 11.8% and 11.1% improvements. BC, BPPO, and PID demonstrate improved performance because only a small part of actions need to be optimized, making it easier to fit compared to high-dimensional actions, while they still show significantly lower performance compared to diffusion-based methods. Visualizations are presented in Appendix G; they show that CL-DiffPhyCon achieves lower error regarding the target state compared with baselines. In terms of inference time besides the enhanced control performance, CL-DiffPhyCon brings significant acceleration of the sampling process, about H/h times faster than DiffPhyCon-h and two times faster than RDM. Furthermore, by combining with DDIM of 30 sampling steps, CL-DiffPhyCon achieves an additional $5\times$ speedup, demonstrating the independent acceleration effect of CL-DiffPhyCon from existing fast sampling methods of diffusion models. As a result, the reduced time cost of CL-DiffPhyCon is comparable with those non-diffusion control baselines BC, BPPO, and PID, while the performance is much superior.

5.3 2D INCOMPRESSIBLE FLUID CONTROL

Experiment settings. Our experimental setup is based on previous studies of Holl et al. (2020); Wei et al. (2024). Given an initial cloud of smoke in a 64×64 incompressible fluid field, this task aims to minimize the volume of smoke failing to pass through the top middle exit (seven exits in total) over $N = 64$ physical time steps, by applying a sequence of 2D forces outside the outermost obstacles. This task represents a simplified scenario of real-life applications such as indoor air quality control (Nair et al., 2022). We consider two settings from real-world consideration: **large domain control**, where control signals are applied to all peripheral regions consisting of 1,792 cells outside the outermost obstacles, and **boundary control**, where control signals are restricted to only the 4×8 cells inside the four exits. It is very challenging since the control forces can only be exerted indirectly in the peripheral regions, which necessitates the model to plan ahead to prevent the smoke from entering the wrong exits or getting stuck in corners. The boundary control setting is even more challenging due to the significantly reduced range of influence of the controllable cells. During inference, for both settings, we follow Zhou et al. (2024) and add $p = 0.1$ probability of random control in the execution of control signals. Random controls cause unexpected changes in the system state, which may render previously planned control sequences no longer applicable, thus necessitating re-planning and adding additional challenge to the task. For both settings, besides a fixed map (FM) evaluation mode where test samples use the same obstacles' configuration with training trajectories, we also introduce a random map (RM) mode where the obstacles' configuration varies. For details of experimental settings and implementation, please refer to Appendix E.

Results. Table 2 shows the control performance of CL-DiffPhyCon and baselines. The results indicate that CL-DiffPhyCon outperforms the baselines in both large domain control and boundary

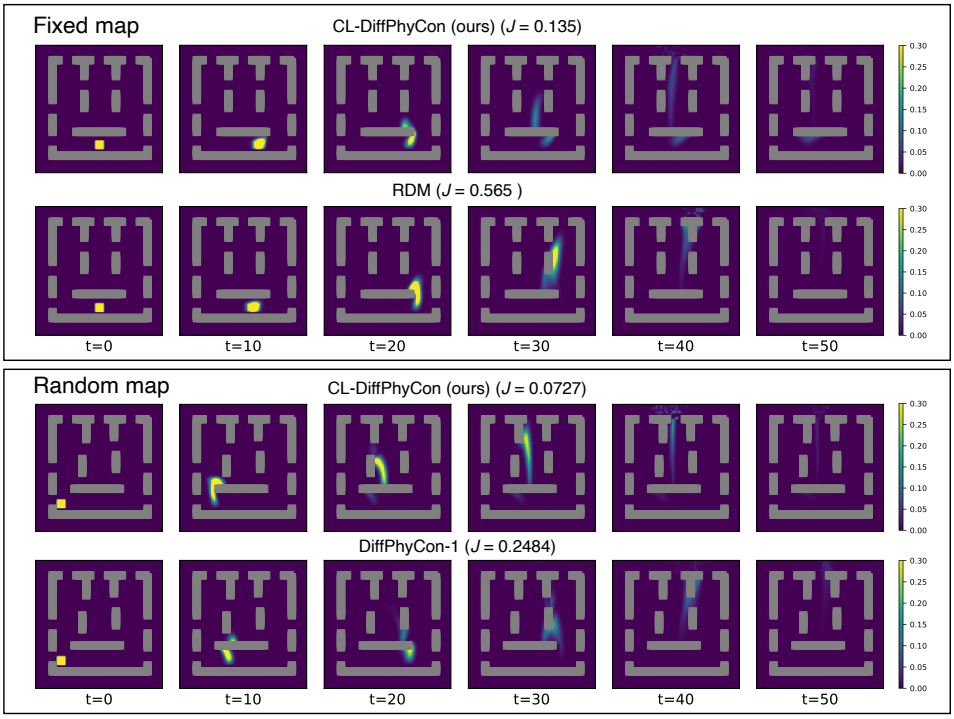

Figure 3: Visual comparisons between CL-DiffPhyCon and best baselines on 2D fluid control under fixed map (top) and random map (bottom) evaluation modes, respectively.

control settings, under both fixed map and random map evaluation modes. This validates that our asynchronous diffusion model could effectively sample appropriate subsequent control sequences by conditioning on the changed system state in a closed-loop manner, showcasing strong generalizability. The results also demonstrate the advantage of diffusion-based control methods over BC and BPPO. In Figure 3, we illustrate randomly selected test samples controlled by CL-DiffPhyCon and the best baselines in the large domain control setting. CL-DiffPhyCon exhibits a stronger capability of adapting to changed fluid and maps. More visualization results are presented in Appendix H. We also present the comparison of average inference time in Table 2. Aligning with our observation on the 1D task, CL-DiffPhyCon achieves approximately H/h times speedup compared to DiffPhyCon-h. Besides, CL-DiffPhyCon is much more efficient than RDM by avoiding frequent replanning. By adopting DDIM, the inference is further accelerated, over $5\times$ faster than strong diffusion-based control baseline RDM, with comparable control performance. For a detailed study of the effect of the model horizon H, please refer to Appendix E.3.

6 Conclusion and Limitation

In this paper, we propose CL-DiffPhyCon, a novel diffusion-based method for closed-loop control of complex physical systems, grounded in a theoretical analysis of the target distribution. Experiments on two physical control tasks demonstrate its superior performance and efficiency. Still, it has several limitations that offer opportunities for future work. First, CL-DiffPhyCon is currently trained offline without interacting with the system dynamics. Incorporating real-time feedback during training could enable dynamic adaptation and the discovery of new strategies. Second, while the two diffusion models in CL-DiffPhyCon are theoretically derived, a formal bound on its optimization performance under guidance sampling remains an open question, providing a promising direction for further theoretical research. Finally, the effectiveness of CL-DiffPhyCon in more domains is worth exploring. Although our method is designed for the control of complex physical systems, it holds promise for broader applications, such as robot control and drone control.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of Westlake University Research Center for Industries of the Future and Westlake University Center for High-performance Computing. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding entities.

REFERENCES

Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B Tenenbaum, Tommi S Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2022.

Karl Johan Åström and Tore Hägglund. The future of pid control. In *Digital Control – Past, present, and future of PID Control*, United States, 2000. Elsevier.

Gerben Beintema, Alessandro Corbetta, Luca Biferale, and Federico Toschi. Controlling rayleigh–bénard convection via reinforcement learning. *Journal of Turbulence*, 21(9-10):585–605, 2020.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=OnD9zGAGT0k`.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

MA Elhawary. Deep reinforcement learning for active flow control around a circular cylinder using unsteady-mode plasma actuators. *arXiv preprint arXiv:2012.10165*, 2020.

Haodong Feng, Yue Wang, Hui Xiang, Zhiyang Jin, and Dixia Fan. How to control hydrodynamic force on fluidic pinball via deep reinforcement learning. *Physics of Fluids*, 35(4), 2023.

Haodong Feng, Dehan Yuan, Jiale Miao, Jie You, Yue Wang, Yi Zhu, and Dixia Fan. Efficient navigation of a robotic fish swimming across the vortical flow field. *arXiv preprint arXiv:2405.14251*, 2024.

Elie Hachem, Hassan Ghraieb, Jonathan Viquerat, Aurélien Larcher, and P Meliga. Deep reinforcement learning for the control of conjugate heat transfer. *Journal of Computational Physics*, 436: 110317, 2021.

Michael Hesse, Julia Timmermann, Eyke Hüllermeier, and Ansgar Trächtler. A reinforcement learning strategy for the swing-up of the double pendulum on a cart. *Procedia Manufacturing*, 24: 15–20, 2018. ISSN 23519789. doi: 10.1016/j.promfg.2018.06.004.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in Neural Information Processing Systems*, 35:8633–8646, 2022.

Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020.

Rakhoon Hwang, Jae Yong Lee, Jin Young Shin, and Hyung Ju Hwang. Solving PDE-Constrained Control Problems Using Operator Learning. *AAAI*, 36(4):4504–4512, June 2022. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v36i4.20373. URL https://ojs.aaai.org/index.php/AAAI/article/view/20373.

Liviu Gr. Ixaru and Guido Vanden Berghe. *Runge-Kutta Solvers for Ordinary Differential Equations*, pp. 223–304. Springer Netherlands, Dordrecht, 2004. ISBN 978-1-4020-2100-8. doi: 10.1007/978-1-4020-2100-8_6. URL https://doi.org/10.1007/978-1-4020-2100-8_6.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *Proceedings of Machine Learning Research*, 162:9902–9915, 17–23 Jul 2022a.

Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 9902–9915. PMLR, 17–23 Jul 2022b.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation*, pp. 1470–1477. IEEE, 2011.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Yun Li, Kiam Heong Ang, and G.C.Y. Chong. Pid control system analysis and design. *IEEE Control Systems Magazine*, 26(1):32–41, 2006. doi: 10.1109/MCS.2006.1580152.

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

Ajith N Nair, Prashant Anand, Abraham George, and Nilabhra Mondal. A review of strategies and their effectiveness in reducing indoor airborne transmission and improving indoor air quality. *Environmental Research*, 213:113579, 2022.

Guido Novati, Siddhartha Verma, Dmitry Alexeev, Diego Rossinelli, Wim M Van Rees, and Petros Koumoutsakos. Synchronisation through learning for two self-propelled swimmers. *Bioinspiration & biomimetics*, 12(3):036001, 2017.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Timo Ewalds, Andrew El-Kadi, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, Remi Lam, and Matthew Willson. Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*, 2023.

Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Réglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of fluid mechanics*, 865:281–302, 2019.

Ricardo Reyes Garza, Nikos Kyriakopoulos, Zoran M Cenev, Carlo Rigoni, and Jaakko VI Timonen. Magnetic quincke rollers with tunable single-particle dynamics and collective states. *Science Advances*, 9(26):eadh2522, 2023.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=a9bzTv9SzO.

Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: An engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5-6):1327–1349, 2021.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv:2010.02502*, October 2020. URL https://arxiv.org/abs/2010.02502.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Siddhartha Verma, Guido Novati, and Petros Koumoutsakos. Efficient collective swimming by harnessing vortices through deep reinforcement learning. *Proceedings of the National Academy of Sciences*, 115(23):5849–5854, 2018.

ZP Wang, RJ Lin, ZY Zhao, X Chen, PM Guo, N Yang, ZC Wang, and DX Fan. Learn to flap: foil non-parametric path planning via deep reinforcement learning. *Journal of Fluid Mechanics*, 984: A9, 2024.

Long Wei, Peiyan Hu, Ruiqi Feng, Haodong Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. A generative approach to control complex physical systems. *arXiv preprint arXiv:2407.06494*, 2024.

Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 1(1):1–34, 2020.

Tailin Wu, Takashi Maruyama, Long Wei, Tao Zhang, Yilun Du, Gianluca Iaccarino, and Jure Leskovec. Compositional generative inverse design. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=wmX0CqFSd7.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

Masaki Yamakita, Kenichi Nonaka, and Katsuhisa Furuta. Swing up control of a double pendulum. In *1993 American Control Conference*, pp. 2229–2233, San Francisco, CA, USA, June 1993. IEEE. ISBN 978-0-7803-0860-2. doi: 10.23919/ACC.1993.4793279. URL https://ieeexplore.ieee.org/document/4793279/.

Cagatay Yildiz, Markus Heinonen, and Harri Lähdesmäki. Continuous-time model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 12009–12018. PMLR, 2021.

Siyuan Zhou, Yilun Du, Shun Zhang, Mengdi Xu, Yikang Shen, Wei Xiao, Dit-Yan Yeung, and Chuang Gan. Adaptive online replanning with diffusion models. *Advances in Neural Information Processing Systems*, 36, 2024.

Zifeng Zhuang, Kun LEI, Jinxin Liu, Donglin Wang, and Yilang Guo. Behavior proximal policy optimization. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3c13LptpIph.

A BASIC PROPERTIES OF STOCHASTIC DIFFERENTIAL EQUATION (SDE)

In this work, we use the formulas of stochastic differential equation (SDE) to express diffusion models. Specifically, the SDEs that we use can be written to the following unified form:

$$\mathrm{d}\mathbf{x}(t) = f(t)\mathbf{x}(t)\mathrm{d}t + g(t)\mathrm{d}\omega(t). \tag{17}$$

The reverse-time SDE of Eq. 17 is:

$$\mathrm{d}\mathbf{x}(t) = [f(t)\mathbf{x}(t) - g(t)^2\nabla_{\mathbf{x}(t)}\log p_t\big(\mathbf{x}(t)\big)]\mathrm{d}t + g(t)\mathrm{d}\omega(t). \tag{18}$$

In this section, we summarize some basic properties of Eq. 17, which can also be found in existing SDE works Song et al. (2021); Chung et al. (2023); Karras et al. (2022).

First, such a linear SDE has the closed solution Särkkä & Solin (2019)

$$\mathbf{x}(t) = \mathbf{x}(0)\exp(\int_0^t f(\eta)\mathrm{d}\eta) + \int_0^t \exp(\int_\zeta^t f(\eta)\mathrm{d}\eta)g(\zeta)\mathrm{d}\omega(\zeta), \tag{19}$$

where the second term is an Itô integral. And the mean $\mathbf{m}(t)$ and covariance $\mathbf{V}(t)$ satisfies the following ordinary differential equations:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{m}(t)}{\mathrm{d}t} &= f(t)\mathbf{m}(t),\\ \frac{\mathrm{d}\mathbf{V}(t)}{\mathrm{d}t} &= 2f(t)\mathbf{V}(t) + g(t)^2\mathbf{I}.\end{aligned} \tag{20}$$

Solving these equations, we get:

$$\begin{aligned}\mathbf{m}(t) &= s(t)\mathbf{m}(0),\\ \mathbf{V}(t) &= s(t)^2(\sigma(t)^2\mathbf{I} + \mathbf{V}(0)),\end{aligned} \tag{21}$$

where $s(t) = \exp\big(\int_0^t f(\eta)\mathrm{d}\eta\big)$ and $\sigma(t) = \sqrt{\int_0^t \frac{g(\eta)^2}{s(\eta)^2}\mathrm{d}\eta}$.

From Eq. 19 and Eq. 21, we know

$$\mathbf{x}(t)|\mathbf{x}(0) \sim \mathcal{N}\big(s(t)\mathbf{x}(0), s(t)^2\sigma(t)^2\mathbf{I}\big). \tag{22}$$

According to the density function of normal distribution, we have:

$$\nabla_{\mathbf{x}(t)}\log p\big(\mathbf{x}(t)|\mathbf{x}(0)\big) = s(t)^{-2}\sigma(t)^{-2}(s(t)\mathbf{x}(0) - \mathbf{x}(t)). \tag{23}$$

Take the expectation over $\mathbf{x}(0)$ conditional on $\mathbf{x}(t)$ on both sides:

$$\begin{aligned}&s(t)^{-2}\sigma(t)^{-2}(s(t)\mathbb{E}[\mathbf{x}(0)|\mathbf{x}(t)] - \mathbf{x}(t))\\ =&\mathbb{E}_{\mathbf{x}(0)}[\nabla_{\mathbf{x}(t)}\log p\big(\mathbf{x}(t)|\mathbf{x}(0)\big)|\mathbf{x}(t)]\\ =&\int \nabla_{\mathbf{x}(t)}\log p\big(\mathbf{x}(t)|\mathbf{x}(0)\big)p\big(\mathbf{x}(0)|\mathbf{x}(t)\big)\mathrm{d}\mathbf{x}(0)\\ =&\int \nabla_{\mathbf{x}(t)}p\big(\mathbf{x}(t),\mathbf{x}(0)\big)/p\big(\mathbf{x}(t)\big)\mathrm{d}\mathbf{x}(0)\\ =&\nabla_{\mathbf{x}(t)}p\big(\mathbf{x}(t)\big)/p\big(\mathbf{x}(t)\big)\\ =&\nabla_{\mathbf{x}(t)}\log p\big(\mathbf{x}(t)\big).\end{aligned} \tag{24}$$

Rearranging the equation, we get Tweedie's estimate:

$$\hat{\mathbf{x}}(0) \triangleq \mathbb{E}[\mathbf{x}(0)|\mathbf{x}(t)] = s(t)^{-1}\mathbf{x}(t) + s(t)\sigma(t)^2\nabla_{\mathbf{x}(t)}\log p\big(\mathbf{x}(t)\big). \tag{25}$$

Now we consider the following loss function Ho et al. (2020):

$$\begin{aligned}&\mathbb{E}_{t,\mathbf{x}(0),\boldsymbol{\epsilon}}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi\big(s(t)\mathbf{x}(0) + s(t)^2\sigma(t)^2\boldsymbol{\epsilon}, t\big)\|_2^2]\\ =&\mathbb{E}_{t,\mathbf{x}(0),\mathbf{x}(t)}[\|\frac{\mathbf{x}(t) - \mathbf{x}(0)s(t)}{s(t)^2\sigma(t)^2} - \boldsymbol{\epsilon}_\phi\big(\mathbf{x}(t), t\big)\|_2^2].\end{aligned} \tag{26}$$

The best prediction $\boldsymbol{\epsilon}_{\phi^*}(\cdot, t)$ in this loss is the following conditional expectation:

$$\begin{aligned}\boldsymbol{\epsilon}_{\phi^*}\big(\mathbf{x}(t), t\big) =& \mathbb{E}[\frac{\mathbf{x}(t) - \mathbf{x}(0)s(t)}{s(t)^2\sigma(t)^2} - \boldsymbol{\epsilon}_{\phi}\big(\mathbf{x}(t), t\big)|\mathbf{x}(t)] \\ =& \frac{\mathbf{x}(t) - \mathbb{E}[\mathbf{x}(0)|\mathbf{x}(t)]s(t)}{s(t)^2\sigma(t)^2}.\end{aligned} \tag{27}$$

Plugging Eq. 25 in it, we get the score function:

$$\nabla_{\mathbf{x}(t)} \log p\big(\mathbf{x}(t)\big) = -\boldsymbol{\epsilon}_{\phi^*}\big(\mathbf{x}(t), t\big). \tag{28}$$

B VARIANCE PRESERVING (VP) SDE

In the practical training and inference, we use Variance Preserving (VP) SDE Song et al. (2021), which is actually the continous version of DDPM Ho et al. (2020).

VP SDE specializes Eq. 17 to

$$\mathrm{d}\mathbf{x}(t) = -\frac{1}{2}\beta(t)\mathbf{x}(t)\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\omega(t), \tag{29}$$

where $\beta(t) > 0$ for $t \in [0, T]$. And its reverse-time SDE is:

$$\mathrm{d}\mathbf{x}(t) = [-\frac{1}{2}\beta(t)\mathbf{x}(t) - \beta(t)\nabla_{\mathbf{x}(t)} \log p_t\big(\mathbf{x}(t)\big)]\mathrm{d}t + \sqrt{\beta(t)}\mathrm{d}\omega(t), \tag{30}$$

In algorithm implementation, we use K time steps, $t_i = \frac{i}{K}T$ for $i = 0, \cdots, K-1$, to discretize Eq. 29. Using $\Delta t = \frac{1}{K}T$, we have:

$$\begin{aligned}\mathbf{x}(t_i + \Delta t) &\approx \mathbf{x}(t_i) - \frac{1}{2}\beta(t_i)\Delta t\mathbf{x}(t_i) + \sqrt{\beta(t_i)\Delta t}\xi, \\ &\approx \sqrt{1 - \beta(t_i)\Delta t}\mathbf{x}(t_i) + \sqrt{\beta(t_i)\Delta t}\xi,\end{aligned} \tag{31}$$

where $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Let $\beta_i = \beta(t_i)\Delta t$, $\mathbf{x}_i = \mathbf{x}(t_i)$. We get the discrete forward iterative equation, which is exact DDPM:

$$\mathbf{x}_{i+1} = \sqrt{1 - \beta_i}\mathbf{x}_i + \sqrt{\beta_i}\xi. \tag{32}$$

Similarly, Eq. 30 can be discretize to

$$\begin{aligned}\mathbf{x}(t_i - \Delta t) &\approx \mathbf{x}(t_i) + \frac{1}{2}\beta(t_i)\Delta t\mathbf{x}(t_i) + \beta(t_i)\Delta t\nabla_{\mathbf{x}(t_i)} \log p_{t_i}\big(\mathbf{x}(t_i)\big) + \sqrt{\beta(t_i)\Delta t}\xi \\ &\approx \frac{1}{\sqrt{1 - \beta(t_i)\Delta t}}\mathbf{x}(t_i) + \beta(t_i)\Delta t\nabla_{\mathbf{x}(t_i)} \log p_{t_i}\big(\mathbf{x}(t_i)\big) + \sqrt{\beta(t_i)\Delta t}\xi.\end{aligned} \tag{33}$$

So, we get the discrete backward iterative equation:

$$\mathbf{x}_{i-1} = \frac{1}{\sqrt{1 - \beta_i}}\mathbf{x}_i + \beta_i\nabla_{\mathbf{x}_i} \log p\big(\mathbf{x}_i\big) + \sqrt{\beta_i}\xi. \tag{34}$$

Additionally, we have:

$$
\begin{aligned}
s(t_i)^2 &= \exp\big(\int_{t_0}^{t_i+\Delta t} -\beta(\eta)\mathrm{d}\eta\big) \\
&= \exp\big(\sum_{j=0}^{i} -\beta(t_j)\Delta t\big) \\
&= \prod_{j=0}^{i} \exp\big(-\beta(t_j)\Delta t\big) \\
&\approx \prod_{j=0}^{i} (1-\beta(t_j)\Delta t) \\
&= \prod_{j=0}^{i} (1-\beta_j).
\end{aligned}
\tag{35}
$$

And, we also have:

$$
\begin{aligned}
s(t_i)^2\sigma(t_i)^2 &= \exp\big(\int_{t_0}^{t_i+\Delta t} -\beta(\eta)\mathrm{d}\eta\big)\int_{t_0}^{t_i+\Delta t} \beta(\eta)\exp\big(\int_{t_0}^{\eta}\beta(\zeta)\mathrm{d}\zeta\big)\mathrm{d}\eta \\
&= \int_{t_0}^{t_i+\Delta t} \beta(\eta)\exp\big(\int_{\eta}^{t_i+\Delta t} -\beta(\zeta)\mathrm{d}\zeta\big)\mathrm{d}\eta \\
&\approx 1-\exp\big(\int_{t_0}^{t_i+\Delta t} -\beta(\eta)\exp\big(\int_{\eta}^{t_i+\Delta t} -\beta(\zeta)\mathrm{d}\zeta\big)\mathrm{d}\eta\big) \\
&\approx 1-\exp\big(\int_{t_0}^{t_i+\Delta t} -\beta(\eta)\mathrm{d}\eta\big) \\
&= 1-s(t_i)^2 \\
&\approx 1-\prod_{j=0}^{i}(1-\beta_j).
\end{aligned}
\tag{36}
$$

We define $\alpha_i = 1-\beta_i$ and $\bar{\alpha}_i = \prod_{j=0}^{i}\alpha_i$. Then, we get $s(t_i)^2 \approx \bar{\alpha}_i$, $s(t_i)^2\sigma(t_i)^2 \approx 1-\bar{\alpha}_i$, and $\mathbf{x}_i|\mathbf{x}_0 \sim \mathcal{N}\big(\sqrt{\bar{\alpha}_i}\mathbf{x}_0, (1-\bar{\alpha}_i)\mathbf{I}\big)$. Therefore, Tweedie's estimate becomes

$$
\hat{\mathbf{x}}(0) = \frac{1}{\sqrt{\bar{\alpha}_i}}(x_i + (1-\bar{\alpha}_i)\nabla_{\mathbf{x}_i}\log p(\mathbf{x}_i)). \tag{37}
$$

According to Eq. 21, the covariance of $\mathbf{x}_i$ is $(1-\bar{\alpha}_i)\mathbf{I} + \bar{\alpha}_i\mathbf{V}_0$. Assuming the clear data has covariance $\mathbf{I}$, we can start from $\mathbf{x}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ during inference.

C DERIVATIONS

Proof of Theorem 1. We first conduct the following decomposition:

$$
\begin{aligned}
&p\big(\mathbf{z}_{1:N}(0), \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \cdots, \tilde{\mathbf{z}}_{N+1:N+H}(\frac{1}{H}T)|\mathbf{u}_0\big)\\
=&p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \mathbf{z}_1(0), \tilde{\mathbf{z}}_{2:H+1}(\frac{1}{H}T), \cdots, \mathbf{z}_N(0), \tilde{\mathbf{z}}_{N+1:N+H}(\frac{1}{H}T)|\mathbf{u}_0\big)\\
=&p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big)\\
&\prod_{\tau=1}^{N} p\big(\mathbf{z}_\tau(0), \tilde{\mathbf{z}}_{\tau+1:\tau+H}(\frac{1}{H}T)|\mathbf{u}_0, \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \mathbf{z}_1(0), \tilde{\mathbf{z}}_{2:H+1}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\\
=&p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big)\\
&\prod_{\tau=1}^{N} p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0), \mathbf{z}_{\tau+H}(T)|\mathbf{u}_0, \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \mathbf{z}_1(0), \tilde{\mathbf{z}}_{2:H+1}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\\
=&p\big(\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T)|\mathbf{u}_0\big) \prod_{\tau=1}^{N} \mathcal{N}(\mathbf{z}_{\tau+H}(T)|\mathbf{0}, \sigma_T^2\mathbf{I})\\
&\prod_{\tau=1}^{N} p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_0, \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \mathbf{z}_1(0), \tilde{\mathbf{z}}_{2:H+1}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big).
\end{aligned}
\tag{38}
$$

The third equation holds since $[\mathbf{z}_\tau(0), \tilde{\mathbf{z}}_{\tau+1:\tau+H}(\frac{1}{H}T)]$ and $[\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0), \mathbf{z}_{\tau+H}(T)]$ are two different arrangements of a same vector $[\mathbf{z}_\tau(0), \mathbf{z}_{\tau+1}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau+H-1}(\frac{H-1}{H}T), \mathbf{z}_{\tau+H}(T)]$. The last equation holds since $\mathbf{z}_\tau(T)$ is independently normally distributed with density $\mathcal{N}(\mathbf{z}_\tau(T)|\mathbf{0}, \sigma_T^2\mathbf{I})$ for any τ. Then, we analyze the conditional probability $p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_0, \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \mathbf{z}_1(0), \tilde{\mathbf{z}}_{2:H+1}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)$ in detail.

Due to Markov property of physical systems, when $\mathbf{u}_{\tau-1}(0)$ appears, other variables with temporal subscripts less than τ can not affect those with subscripts greater than or equal to τ. Therefore, we can remove those variables with subscripts less than τ, except for $\mathbf{u}_{\tau-1}(0)$. Thus, we have:

$$
\begin{aligned}
&p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_0, \tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \mathbf{z}_1(0), \tilde{\mathbf{z}}_{2:H+1}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\\
=&p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \mathbf{c}, \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big),
\end{aligned}
\tag{39}
$$

where $\mathbf{c}$ denotes the subset of latent variables $[\mathbf{z}_\tau(T), \tilde{\mathbf{z}}_{\tau:\tau+1}(\frac{H-1}{H}T), \cdots, \tilde{\mathbf{z}}_{\tau:\tau+H-2}(\frac{2}{H}T)]$ extracted from $[\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T), \cdots, \mathbf{z}_{\tau-1}(0), \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)]$. Further, by adding new variables $\mathbf{z}_{\tau:\tau+H-1}(0)$ to this probability, we have

$$
\begin{aligned}
&p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \mathbf{c}, \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\\
=&\int p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \mathbf{c}, \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T), \mathbf{z}_{\tau:\tau+H-1}(0)\big)\\
&p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0), \mathbf{c}, \tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\mathrm{d}\mathbf{z}_{\tau:\tau+H-1}(0).
\end{aligned}
\tag{40}
$$

The first distribution inside the integral in Eq. 40 can be simplified as follows:

$$
\begin{aligned}
&p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)\big)\\
=&\frac{p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0),\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)}{p\big(\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)}\\
=&\frac{p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)p\big(\mathbf{c}|\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)}{p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)p\big(\mathbf{c}|\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)}\\
=&\frac{p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)}{p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)}\\
=&p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)\big).
\end{aligned}
\tag{41}
$$

In the second equation, we use the Markov property of the diffusion process of $\mathbf{z}_\tau(t)$ over t, and skip a step including the following two equations

$$
\begin{aligned}
p\big(\mathbf{c}|\mathbf{u}_{\tau-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)&=p\big(\mathbf{c}|\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\\
p\big(\mathbf{c}|\mathbf{u}_{\tau-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)&=p\big(\mathbf{c}|\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)
\end{aligned}
$$

in the numerator and denominator respectively of the right side of the second equation. Similarly, we show that we can also simplify the other distribution inside the integral in Eq. 40.

$$
\begin{aligned}
&p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\\
=&\frac{p\big(\mathbf{z}_{\tau:\tau+H-1}(0),\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0)\big)}{p\big(\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0)\big)}\\
=&\frac{p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)p\big(\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0)\big)}{p\big(\mathbf{c},\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0)\big)}\\
=&\frac{p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0)\big)p\big(\mathbf{c}|\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)}{p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0)\big)p\big(\mathbf{c}|\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)}\\
=&\frac{p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0)\big)}{p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)|\mathbf{u}_{\tau-1}(0)\big)}\\
=&p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big).
\end{aligned}
\tag{42}
$$

Combining Eq. 39 to Eq. 42, we get the following result:

$$
\begin{aligned}
&p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_0,\tilde{\mathbf{z}}_{1:H}(\frac{1}{H}T),\mathbf{z}_1(0),\tilde{\mathbf{z}}_{2:H+1}(\frac{1}{H}T),\cdots,\mathbf{z}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\\
=&\int p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T),\mathbf{z}_{\tau:\tau+H-1}(0)\big)\\
&p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\mathrm{d}\mathbf{z}_{\tau:\tau+H-1}(0)\\
=&\int p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big)\mathrm{d}\mathbf{z}_{\tau:\tau+H-1}(0)\\
=&p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0),\tilde{\mathbf{z}}_{\tau:\tau+H-1}(\frac{1}{H}T)\big).
\end{aligned}
\tag{43}
$$

Plugging this to Eq. 38, we prove the conclusion. □

Proof of Proposition 1. Adding new variables $\mathbf{z}_{\tau:\tau+H-1}(0)$, we have:

$$\begin{aligned}&p_t\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)\big)\\=&\int p\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0)\big)p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)\mathrm{d}\mathbf{z}_{\tau:\tau+H-1}(0).\end{aligned} \tag{44}$$

Notice that the distributions of $\{\mathbf{z}_{\tau+i}(t+\frac{i}{H}T)\}_{i=0}^{H-1}$ are conditional independent. Therefore, we can reformulate the above equation to:

$$\begin{aligned}&p_t\big(\tilde{\mathbf{z}}_{\tau:\tau+H-1}(t)|\mathbf{u}_{\tau-1}(0)\big)\\=&\int\prod_{i=0}^{H-1}p\big(\mathbf{z}_{\tau+i}(t+\frac{i}{H}T)|\mathbf{u}_{\tau-1}(0),\mathbf{z}_{\tau:\tau+H-1}(0)\big)p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)\mathrm{d}\mathbf{z}_{\tau:\tau+H-1}(0)\\=&\int\prod_{i=0}^{H-1}p\big(\mathbf{z}_{\tau+i}(t+\frac{i}{H}T)|\mathbf{z}_{\tau+i}(0)\big)p\big(\mathbf{z}_{\tau:\tau+H-1}(0)|\mathbf{u}_{\tau-1}(0)\big)\mathrm{d}\mathbf{z}_{\tau:\tau+H-1}(0)\\=&\mathbb{E}_{\mathbf{z}_{\tau:\tau+H-1}(0)}[\prod_{i=0}^{H-1}p\big(\mathbf{z}_{\tau+i}(t+\frac{i}{H}T)|\mathbf{z}_{\tau+i}(0)\big)].\end{aligned} \tag{45}$$

Here, the second equation is because $\mathbf{z}_{\tau+i}(t+\frac{i}{H}T)$ only depends on $\mathbf{z}_{\tau+i}(0)$ with the distribution Eq. 22 described in Section A. And we have proved the desired conclusion. □

Proof of Proposition 2. This proposition is just plugging the $p_{\mathrm{G}}\big(\mathbf{u}_{\text{env},\tau}|\mathbf{u}_{\text{env},\tau-1},\mathbf{w}_\tau(0)\big)$ to the joint distribution in Proposition 1. And we omit this trivial proof. □

D Details of 1D Burgers' equation control

D.1 Dataset

We follow instructions in Wei et al. (2024) to generate a 1D Burgers' equation dataset. Specifically, we use the finite difference method (FDM) to generate trajectories in a domain of space range $x \in [0,1]$ and time range $\tau \in [0,1]$, with random initial states and control sequences following certain distributions. The space is discretized into 128 cells and time into 10000 steps. We generated 90000 trajectories for the training set and 50 for the testing set. For each training sample, its target state $u_d(\tau,x)$ is randomly selected from other training samples, which means that almost all the samples in the training set are unsuccessful as they could hardly achieve the target states under random controls. Thus it is challenging to generate control sequences with performance beyond the training dataset.

D.2 Experimental Settings

Similar to (Wei et al., 2024), we design different settings of 1D Burgers' equation control in Section 5.2:

Noise-free: This is a scenario where all states $u(\tau,x), x \in [0,1]$ for $\tau \in [0,1]$ of the system can be observed. The system, control, and measurement do not have the noise.

Physical constraint: In real-world scenarios, the actuator often has the upper and lower limits due to the physical constraints. We limit the control with bounds -2 to 2 in this setting, while unconstrained control ranges from -5 to 5.

System noise: For complex engineering systems, the practical physical systems often have noise issues in the system (plant) and actuator (control). We consider such real-world scenarios and perturb the system by Gaussian noise with standard deviation $\sigma = 0.025$ following the work (Yildiz et al., 2021). It can also simulate the control noise setting as the noise added to the external force can be

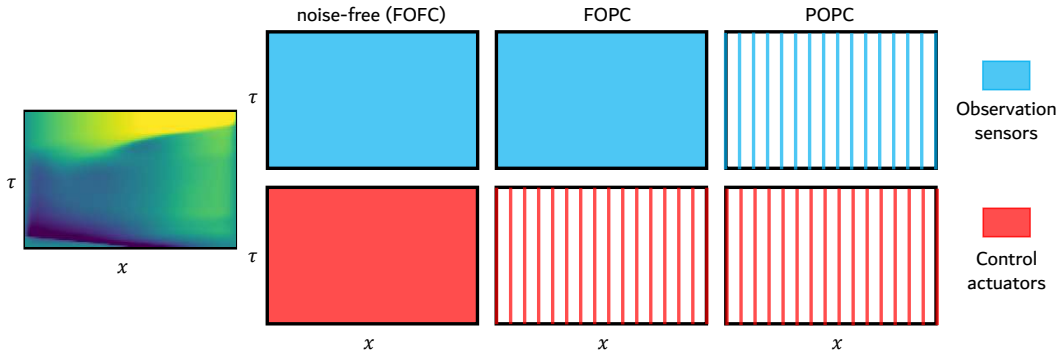

Figure 4: **Illustration of partial observation and control.** Blue represents the position observed by sensors, and red represents the position that actuators are able to control. The region filled entirely with blue and red indicates that all spatial regions can be observed and controlled, respectively. The vertical line indicates where there are sensors and actuators in that spatial position.

decomposed and considered as the system noise for the 1D Burgers' equation in Eq. 15. Specifically, for a deterministic system: Burgers' equation, $G_0 : \frac{\partial u}{\partial \tau} = -u \cdot \frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2} + w(\tau, x)$, where u denotes the state variable and w represents the control. To account for system noise, we augment the dynamics with additive Gaussian noise: $G_1 : \frac{\partial u}{\partial \tau} = -u \cdot \frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2} + w(\tau, x) + \xi_\tau, \xi_\tau \sim \mathcal{N}(0, \sigma)$. During numerical simulation, the perturbed system (G_1) supersedes the nominal system (G_0) when implementing the control input w. Consequently, the feedback state u is obtained as the solution to the stochastic PDE (G_1).

Measurement noise: The measurement noise is also a common phenomenon in practical physical systems. We consider the measurement with Gaussian noise. The feedback state is characterized by the superposition of the nominal solution u to system (G_0) and a Gaussian noise term $\xi_\tau \sim \mathcal{N}(0, \sigma)$ ($\sigma = 0.025$ in the experiments), thereby incorporating measurement noise into the feedback loop while preserving the dynamics of the nominal system.

Partial observations and partial control: In general, the sensors for observation and actuators for control are located in a small part of the spatial domain, unlike the full observation and full control (FOFC) noise-free setting. In this setting, we consider two observation cases: full observation and 16 sensors observation (1/8 of the spatial domain). Both cases are controlled by 16 evenly placed actuators. Although only partial observations are available, the reported results are calculated for an entire spatial domain, including unobservable parts. (As a common single-input single-output (SISO) controller, the PID controller is difficult to directly apply to multiple-input multiple-output (MIMO) systems. Its application to the MIMO system often requires additional decoupling and target planning modules, and the control effect of PID is greatly influenced by decouplers and planners, making it difficult to compare fairly. Therefore, we only apply the PID in 16 sensors for observation and 16 actuators for control.) The placement of sensors and actuators is illustrated in Figure 4.

Half domain: In this setting, we hide some parts of $\mathbf{u}$ and measure the $\mathcal{J}$ of model control as the white area with oblique lines shown in Figure 11. Specifically, $\mathbf{u}(\tau, x),\ x \in [\frac{1}{4}, \frac{3}{4}]$ is set to zero in the dataset during training and $\mathbf{u}_0(x),\ x \in [\frac{1}{4}, \frac{3}{4}]$ is also set to zero during testing. Only $\Omega = [0, \frac{1}{4}] \cup [\frac{3}{4}, 1]$ is observed, controlled and evaluated.

D.3 Implementation

We use U-Net (Ronneberger et al., 2015) as architectures for the models $\boldsymbol{\epsilon}_\phi$ and $\boldsymbol{\epsilon}_\theta$. Two models are separately trained using the same training dataset. Note that in the partial observation settings, the unobserved data is invisible to the model during both training and testing as introduced in Appendix D.2. We simply pad zero in the corresponding locations of the model input and conditions, and also exclude these locations in the training loss. Therefore, the model only learns the correlation between the observed states and control sequences. We use the MSE loss to train the models. Both models

Table 3: **Hyperparameters of models and training for 1D Burgers' equation control.**

Hyperparameter name	Value
U-Net $\boldsymbol{\epsilon}_\phi(\mathbf{w})$	
Model horizon H	16
Initial dimension	64
Downsampling/Upsampling layers	4
Convolution kernel size	3
Dimension multiplier	$[1, 2, 4, 8]$
Attention hidden dimension	32
Attention heads	4
U-Net $\boldsymbol{\epsilon}_\theta(\mathbf{u}, \mathbf{w})$	
Model horizon H	16
Initial dimension	64
Downsampling/Upsampling layers	4
Convolution kernel size	3
Dimension multiplier	$[1, 2, 4, 8]$
Attention hidden dimension	32
Attention heads	4
Training	
Training batch size	16
Optimizer	Adam
Learning rate	1e-4
Training steps	190000
Learning rate scheduler	cosine annealing
Inference	
Synchronously sampling steps	900
Each asynchronously sampling step	60

Table 4: The average control objective $\mathcal{J}$ of 1D Burgers' equation control in the half domain, and inference time on a single NVIDIA A100 80GB GPU with 16 CPU cores are reported. Bold font is the best model and the runner-up is underlined.

	$\mathcal{J} \downarrow$	time (s) $\downarrow$
BC	0.2558	**0.7150**
BPPO	0.2033	0.7342
PID	0.2212	<u>0.7236</u>
DiffPhyCon-1	0.0196	39.9720
DiffPhyCon-5	0.0184	8.4481
DiffPhyCon-15	0.0192	3.8935
RDM	0.0196	9.8153
CL-DiffPhyCon (ours)	**0.0090**	9.7516
CL-DiffPhyCon (DDIM, ours)	<u>0.0104</u>	0.7628

have $T = 900$ diffusion steps. The DDIM (Song et al., 2020) sampling we use only has 30 diffusion steps with the hyperparameter $\eta = 1$. Hyperparameters of models and training are listed in Table 3.

D.4 Results of half domain

In this subsection, we consider another noise-free setting that only observe, control, and evaluate on half of the spatial domain. From the results in Table 4, our conclusion is consistent with the noise-free scenario, and CL-DiffPhyCon shows significant improvements in both performance and efficiency.

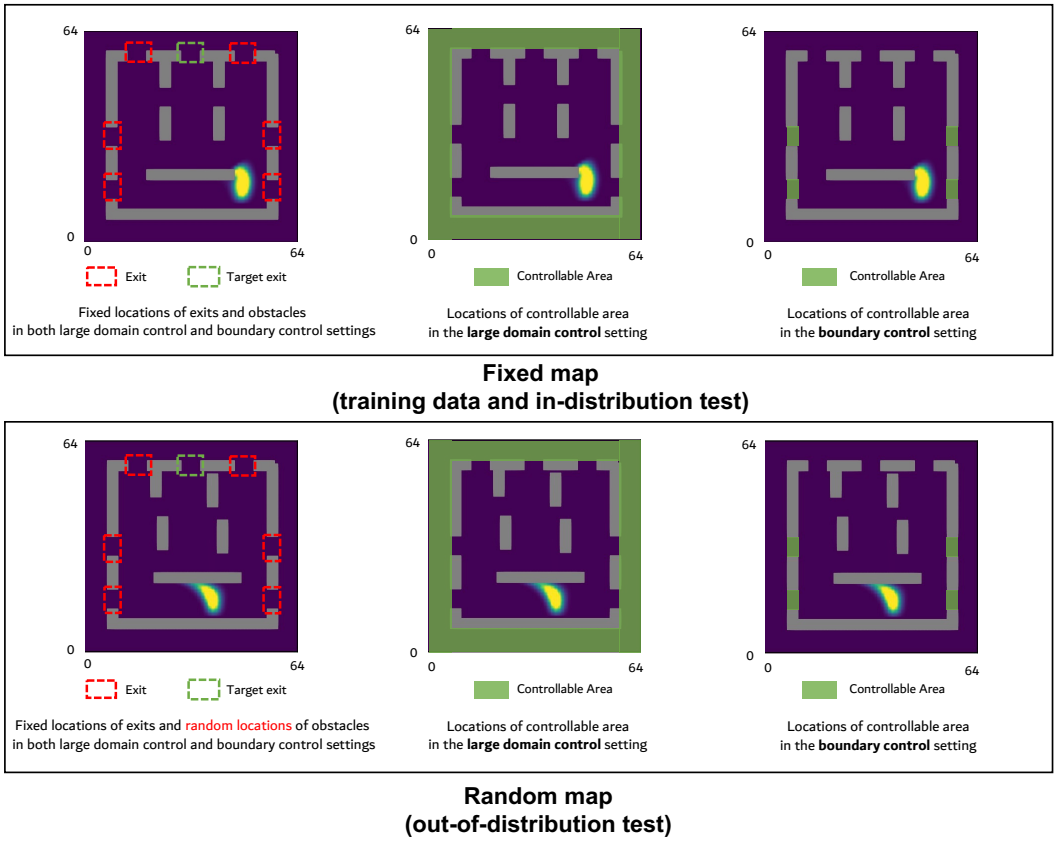

Figure 5: Illustration of the 2D incompressible fluid control task settings. In the **large domain control** setting (middle), control signals are applied to the peripheral green regions surrounding the obstacles. In the **boundary control** setting (right), control signals are limited to the green cells within the four exits. Two evaluation modes are used: **fixed map** (top) for in-distribution testing and **random map** (bottom) for out-of-distribution testing.

Table 5: **Hyperparameters of 2D experiments**.

Hyperparameter Name	Value
Number of attention heads	4
Kernel size of conv3d	(3, 3, 3)
Padding of conv3d	(1,1,1)
Stride of conv3d	(1,1,1)
Kernel size of downsampling	(1, 4, 4)
Padding of downsampling	(1, 2, 2)
Stride of downsampling	(0, 1, 1)
Kernel size of upsampling	(1, 4, 4)
Padding of upsampling	(1, 2, 2)
Stride of upsampling	(0, 1, 1)

E DETAILS OF 2D INCOMPRESSIBLE FLUID CONTROL

E.1 EXPERIMENTAL SETTING

Dynamics of 2D incompressible fluid follows the Navier-Stokes equations:

$$\begin{cases} \frac{\partial \mathbf{v}}{\partial \tau} + \mathbf{v} \cdot \nabla \mathbf{v} - \nu \nabla^2 \mathbf{v} + \nabla p = f, \\ \nabla \cdot \mathbf{v} = 0, \\ \mathbf{v}(0, \mathbf{x}) = \mathbf{v}_0(\mathbf{x}), \end{cases} \tag{46}$$

where f denotes the external force, p denotes pressure, ν denotes the viscosity coefficient and $\mathbf{v}$ denotes velocity. $\mathbf{v}_0(\mathbf{x})$ is the initial condition. We follow the setup of the 2D incompressible fluid control task as described in (Wei et al., 2024), using the Phiflow solver Holl et al. (2020) to simulate fluid dynamics. The resolution of the 2D flow field is set to 64×64, and the flow field is unbounded. We consider two settings: **large domain control** and **boundary control**. In the large domain control setting, control signals are applied to all peripheral regions outside the obstacles, consisting of 1,792 cells, as highlighted in green in the middle subfigures of Figure 5. This setting is consistent with (Wei et al., 2024). In the boundary control setting, control signals are restricted to only the 4×8 cells inside the four exits, as shown in green in the right subfigures of Figure 5. This setting is newly designed in this paper. Both configurations represent indirect control, as the smoke primarily moves within the gray obstacles. The boundary control setting is more challenging due to the significantly reduced number of controllable cells. We also follow (Wei et al., 2024) to generate the training dataset. For both settings, the obstacle locations (gray cells) are shown in the top row of Figure 5 and are consistent across all training trajectories. Each trajectory contains $N = 64$ physical time steps and includes features such as horizontal and vertical velocities, smoke density, and horizontal and vertical control forces. We generated 40,000 training trajectories for the large domain control setting, and 30,000 for the boundary control setting.

During inference, following RDM (Zhou et al., 2024), we add a small level of control noise to make the task more challenging. Specifically, for a trajectory of length $N = 64$, the control signal in each physical time step is executed with probability $p = 0.1$ as a random control, where the horizontal and vertical components following the uniform distributions bounded by the minimal and maximal values of horizontal and vertical control signals in the training dataset, respectively. On each setting of large domain control and boundary control, we design two evaluation modes, fixed map (FM) and random map (RM), to test the generalization capability of each method:

Fixed map (FM): In this mode, all 50 test samples use the same obstacle configuration with training trajectories. However, the initial locations of test samples are different. In this mode, the testing and training initial conditions follow the same distribution (in-distribution test).

Random map (RM): In this mode, the 50 test samples use random obstacles' configuration in the fluid field. Specifically, the movement ranges for the five internal obstacles, whose default locations are shown in the top two rows in Figure 5, are as follows:

- The two upper obstacles can move downward, left, or right by no more than 3 grid spaces.
- The two middle obstacles can move upward, downward, left, or right by no more than 3 grid spaces.
- The one lower obstacle can move upward, left, or right by no more than 3 grid spaces.

In this mode, the testing and training initial conditions follow different distributions (out-of-distribution test). Note that although the obstacles' configuration varies across different test samples, for each test sample, its obstacles' configuration remains unchanged during the control process.

E.2 IMPLEMENTATION

We use 3D U-net Ho et al. (2022) as architectures for the models $\boldsymbol{\epsilon}_\phi$ and $\boldsymbol{\epsilon}_\theta$. Two models are separately trained using the same training dataset. We use the MSE loss to train the models. Both models have $T = 600$ diffusion steps. The DDIM (Song et al., 2020) sampling we use has 75 diffusion steps with the hyperparameter $\eta = 0.3$ for the large domain control setting and 120 diffusion steps with the hyperparameter $\eta = 0.3$ for the boundary control setting. Hyperparameters of models and training are listed in Table 5.

Table 6: Effect of the model horizon H on 2D incompressible fluid control under the large domain control setting and fixed map (FM) evaluation mode. The average control objective $\mathcal{J}$ and inference time on a single NVIDIA A6000 48GB GPU with 16 CPU cores are reported. Bold font is the best model and the runner-up is underlined. The results of $H = 15$ are copied from Table 2 for comparison.

	$H = 6$		$H = 10$		$H = 15$	
	$\mathcal{J} \downarrow$	time (s) $\downarrow$	$\mathcal{J} \downarrow$	time (s) $\downarrow$	$\mathcal{J} \downarrow$	time (s) $\downarrow$
DiffPhyCon-1	<u>0.8986</u>	<u>1022</u>	<u>0.5140</u>	<u>1216</u>	<u>0.5454</u>	<u>1677</u>
CL-DiffPhyCon (ours)	**0.6012**	**143.32**	**0.3367**	**136.81**	<u>0.3371</u>	**140.83**

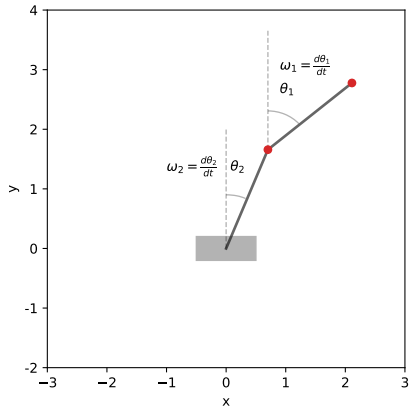

Figure 6: Illustration of the cart-double pendulum system. The angles are measured with respect to the vertical direction, and the angular velocities are defined as the time derivative of the angles.

E.3 Effect of Model Horizon

The choice of the model horizon H is determined by balancing efficiency and effectiveness. We conducted experiments on the 2D incompressible fluid control task under the fixed map (FM) setting with $H = 6$ and $H = 10$. The results are shown in Table 6, where we also copy the results of $H = 15$ from 2 together for comparison.

The results indicate that both our CL-DiffPhyCon and DiffPhyCon-1 yield similar performance when $H = 10$, compared to $H = 15$ as reported in Table 2. However, performance significantly deteriorates when H decreases to 6, which is likely due to the shorter observation window leading to inaccurate future control objective $\mathcal{J}$ estimation and suboptimal guidance sampling (Line 5 in Algorithm 1). On the other hand, increasing H beyond 15 significantly raises GPU memory costs during inference, thus not recommended. Across different values of H (i.e., H=6, 10, and 15), our method consistently outperforms DiffPhyCon-1. Therefore, for this task, a horizon between 10 and 15 is appropriate. In practical applications, the optimal model horizon can be determined through multiple trials to balance performance and efficiency, similar to the approach used in diffusion policies (Chi et al., 2023) (see Figure 5 (left) in the referenced paper).

F Double Pendulum Control Experiment

To investigate the performance of our CL-DiffPhyCon in lower-dimensional control problems, we performed experiments on controlling the system of a cart-inverted double pendulum system. The goal is to keep the double pendulum from falling down by exerting a force on the cart.

F.1 Simulation Environment Setup

The system consists of two point masses fixed on the end of two massless rigid rods, which are mutually connected and connected to a cart with frictionless hinges, as shown in Figure 6. The

cart-double pendulum system is simulated with the 4th order Runge-Kutta method (Ixaru & Vanden Berghe, 2004). In our experiment, the simulation time step interval is set to 5e-3 to ensure the accuracy of the simulation.

The physical quantities (unitless) are as follows: both rods have a length of 0.2, both point masses have a mass of 0.1, the mass of the cart is 1, and the gravitational acceleration is 9.8. The control force is restricted in [-10, 10]. Although the control signal and system states are relatively low-dimensional, we note the control task is non-trivial. Since no torque can be exerted on the hinges, the system can exhibit limited controllability. Besides, the rods deviate from the equilibrium point, and the non-linear or even chaotic dynamics impose significant challenges to its control. This system has been studied in the control research community (Hesse et al., 2018; Yamakita et al., 1993).

We would emphasize that this experiment provides a more easy-to-run verification of our method, as opposed to our 1D Burgers' equation experiment and 2D Navier-Stokes equation control experiments. Our cart-double pendulum environment can run at a refreshing rate of over 300 Hz on an Intel Xeon Platinum 8358 CPU, which allows real-time simulation (200 Hz refreshing rate in our setting) without high-end GPUs need.

F.2 Training and Evaluation Details

In this experiment, we focus on an offline learning setting, where a training dataset containing expert action-state sequences is provided. The training dataset consists of 1e5 trajectories generated by an expert policy pretrained with PPO. Our CL-DiffPhyCon and baseline methods are trained with shared hyperparameters. In all diffusion models, VP-SDE training and ODE sampling are adopted (Song et al., 2021). All methods are trained for 2.9e5 steps, and 16-step Midpoint ODE solver (Karras et al., 2022) is used. The backbone is kept to an identically configured Transformer network. Other hyperparameters are also kept the same.

Our control task is defined on finite-length episodes, with a total of 1 second (200 steps) environment time. We use the metric *success rate* to reflect control performance, which is defined as the ratio of not-falling pendulums when the episode terminates. The not-falling criterion is defined as the upper point height larger than 0.9 maximum height. An ensemble of 100 cart-double pendulum systems is independently randomly initialized and controlled to fairly evaluate the performance of each task. The average and standard deviation are reported for five differently seeded runs to demonstrate the uncertainty of the evaluation.

To mimic realistic systems where the system state may be randomly perturbated, we introduce stochasticity into our cart-double pendulum system. At each step, the environment will perturb the system state (including the angle and angular velocities of two rods, the position, and the velocity of the cart) with probability. Once added, the perturbation will be sampled from a uniform distribution in the range $[-0.02, 0.02]$. To make a comprehensive comparison, we report the results with three different rates: 0, 0.1, and 0.3.

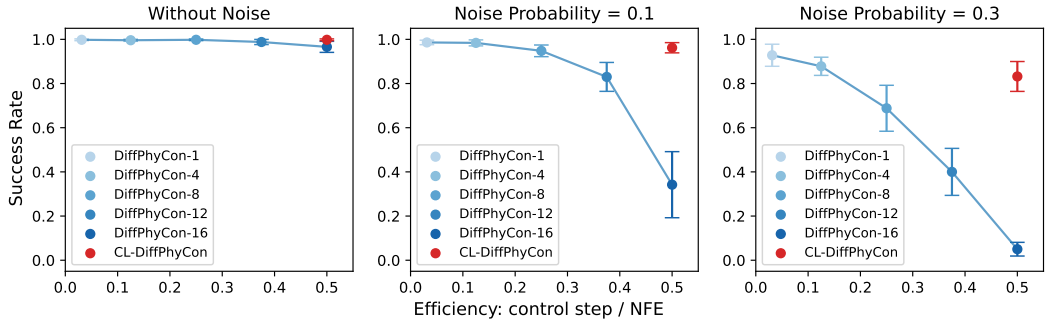

Figure 7: Pareto plot comparing our CL-DiffPhyCon and baselines. The error bar shows $\pm$ 1 standard deviation across 5 seeded runs.

F.3 RESULTS

As shown in Figure 7, CL-DiffPhyCon achieves a remarkable balance between the control algorithm runtime and the control performance. Compared to the baselines of DiffPhyCon-15, DiffPhyCon-12, and DiffPhyCon-8, our CL-DiffPhyCon provides much higher control performance while only requiring similar neural function evaluations (NFE, as in Song et al. (2021); Karras et al. (2022)) per environment step. Although DiffPhyCon-4 and DiffPhyCon-1 show marginal performance improvement over our CL-DiffPhyCon, they are nearly one order of magnitude slower.

For a more intuitive demonstration of the performance comparison, we include a comprehensive visualization where our method is compared with two selected baselines under three different random perturbation settings, as shown in Figure 8. Compared to DiffPhyCon-15, Our CL-DiffPhyCon is able to stabilize the inverted double pendulum for a longer time (subfigure (a)), and the final upper mass state (θ_2, ω_2) is closer to the equilibrium point $(0, 0)$ (subfigure (b)). Additionally, although the performance of our CL-DiffPhyCon is similar to that of DiffPhyCon-1, CL-DiffPhyCon costs about only $1/15$ computation time of it.

G 1D VISUALIZATION RESULTS

We present the visualization results of our method and baselines under noise-free and half domain settings in Figure 9, 10, 11, and 12, respectively. Under each setting, we present the results of four randomly selected samples from the test set.

H 2D VISUALIZATION RESULTS

More visualization results of our method and compared baselines are presented. For both fixed map (Figure 13 and Figure 14) and random map (Figure 15 and Figure 16) settings, we present two randomly selected test samples. Our method outperforms baselines on average. Note that the fluid dynamics is very sensitive to the generated control signals. Thus different methods may perform dramatically differently on the same test sample. However, we find that each method performs stably on each test sample through multiple evaluations.

I BASELINES

For RDM (Zhou et al., 2024), it involves two hyperparameters to determine when to update the previously planned control sequences or replan from scratch. Their default values do not work well on our 1D and 2D tasks. On the 1D task, they are too small, which results in very expensive replanning from scratch almost every physical time, although its performance is still unsatisfactory. On the 2D task, they are too large, leading to few replannings and inferior results. Therefore, we run extensive evaluations to search for their best values. On the 1D task, we find that the best pair of values for the two hyperparameters is (0.06, 0.1) in the noise-free setting. On the 2D task, we find that the best pair of values is (0.0005, 0.002).

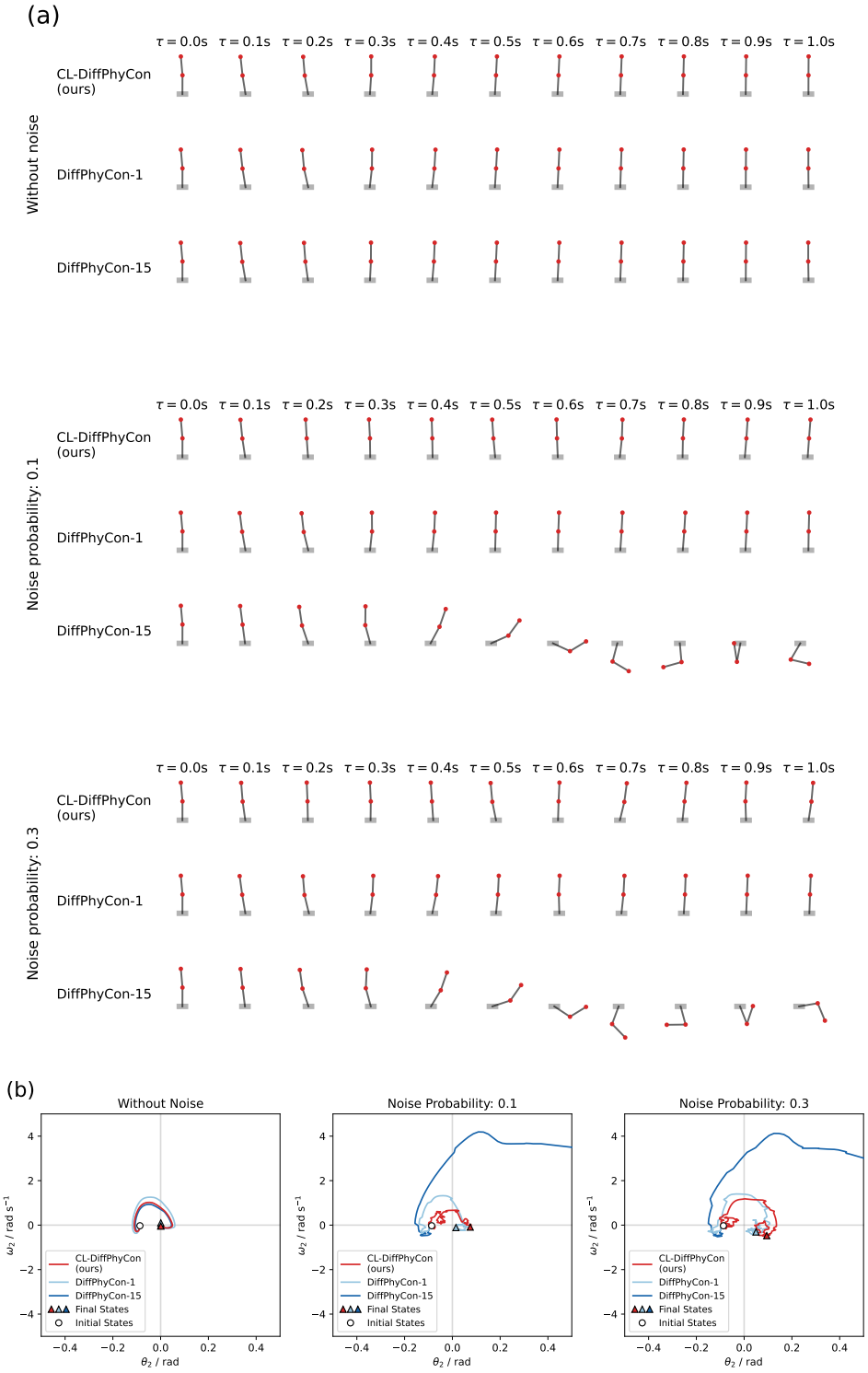

Figure 8: **Visualization of and instance of the cart-double pendulum under control**. The goal is to keep the double pendulum from falling down by exerting a force on the cart over 1 second (200 control steps) sequentially. Three control methods are applied under three different perturbation rates. In each setting, 10 snapshots of the pendulum are shown in subfigure (a), whereas the corresponding trajectories in the phase space are shown in subfigure (b).

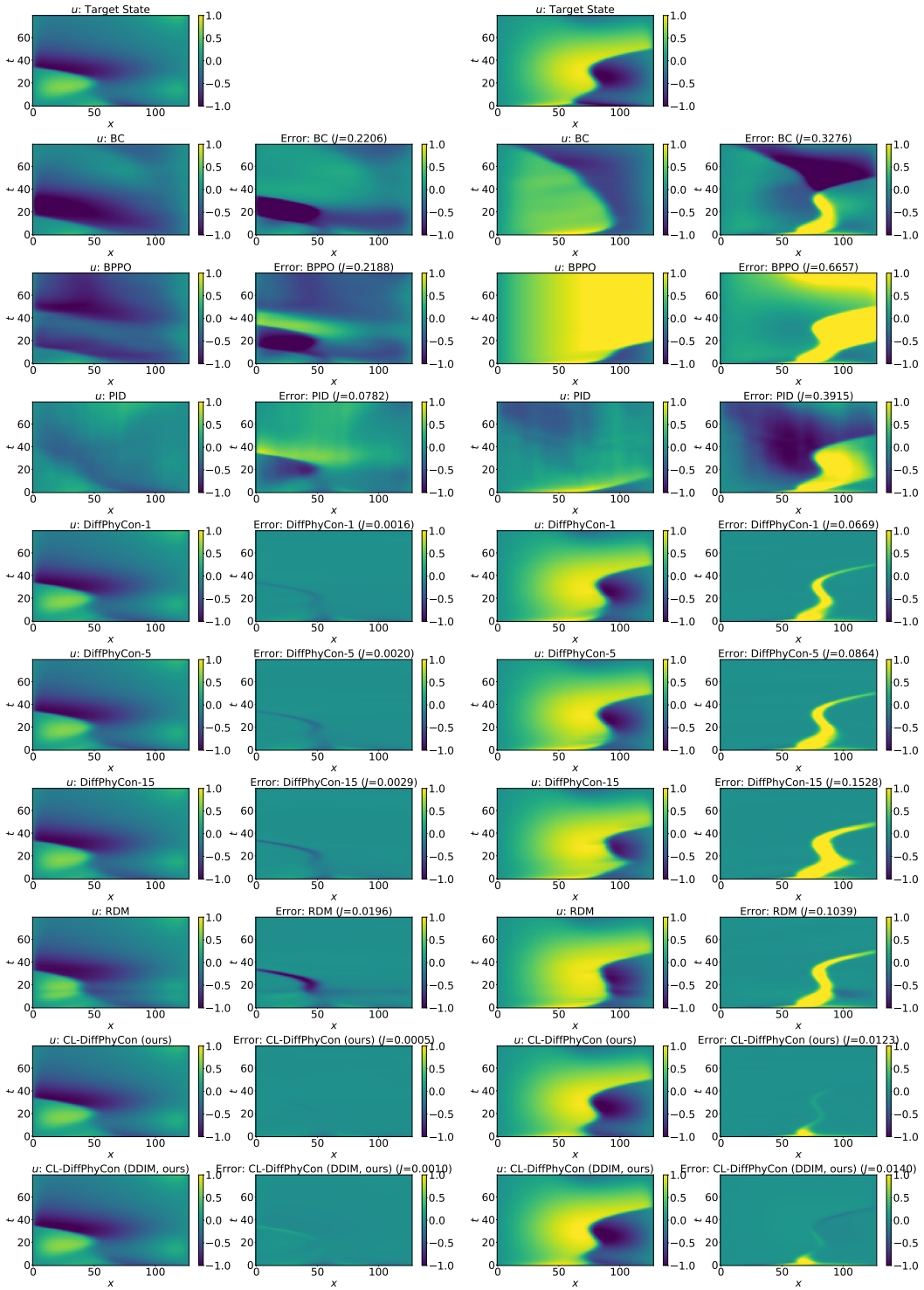

Figure 9: **Two visualizations results of 1D Burgers' equation control under the noise-free setting**. The first line is the target $u_d(t, x)$ and the error $u(t, x) - u_d(t, x)$ measures the gap between the state under control and target. The horizontal axis is the time coordinate and the vertical axis is the state.

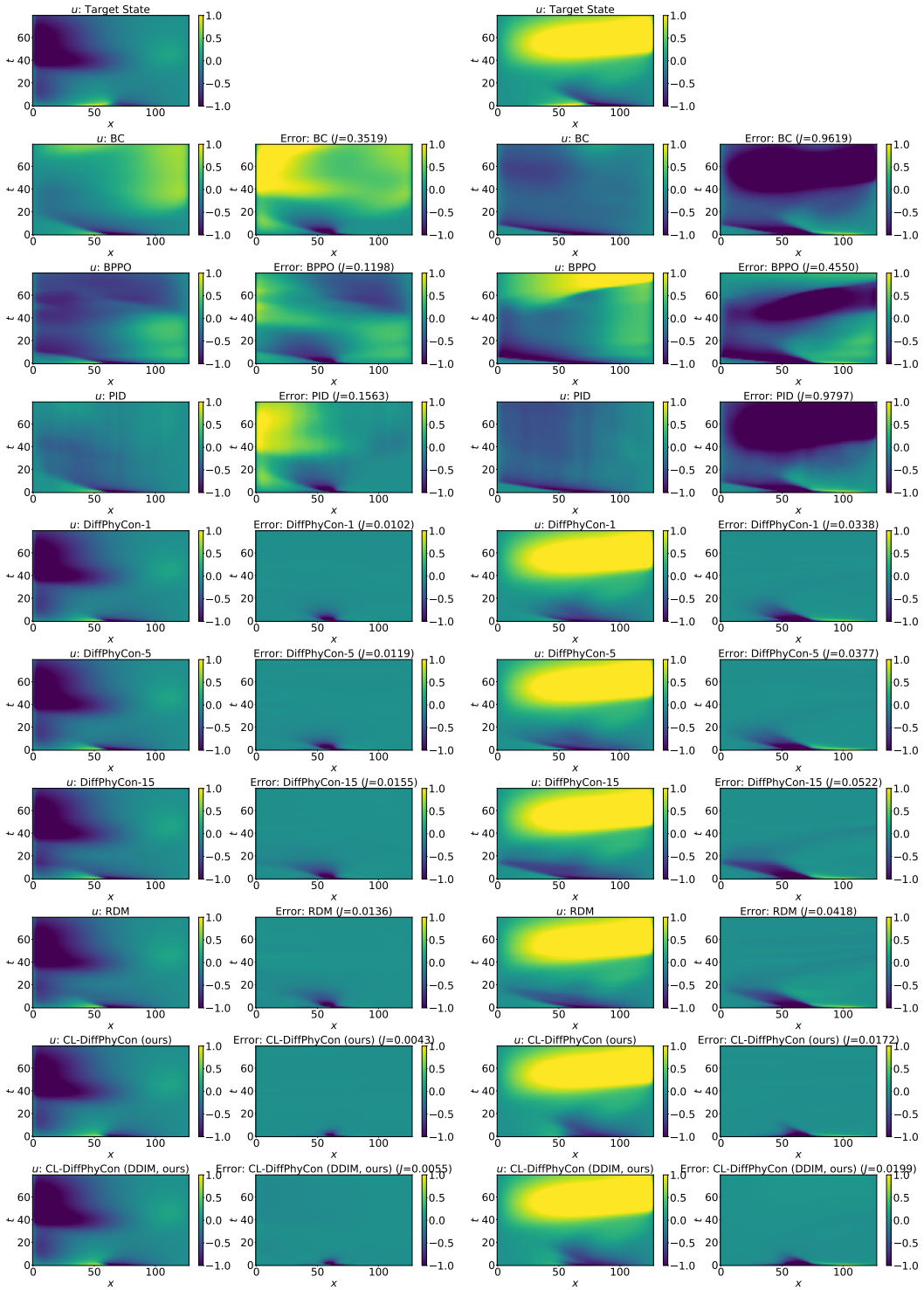

Figure 10: **Two visualizations results of 1D Burgers' equation control under the noise-free setting**. The first line is the target $u_d(t, x)$ and the error $u(t, x) - u_d(t, x)$ measures the gap between the state under control and target. The horizontal axis is the time coordinate and the vertical axis is the state.

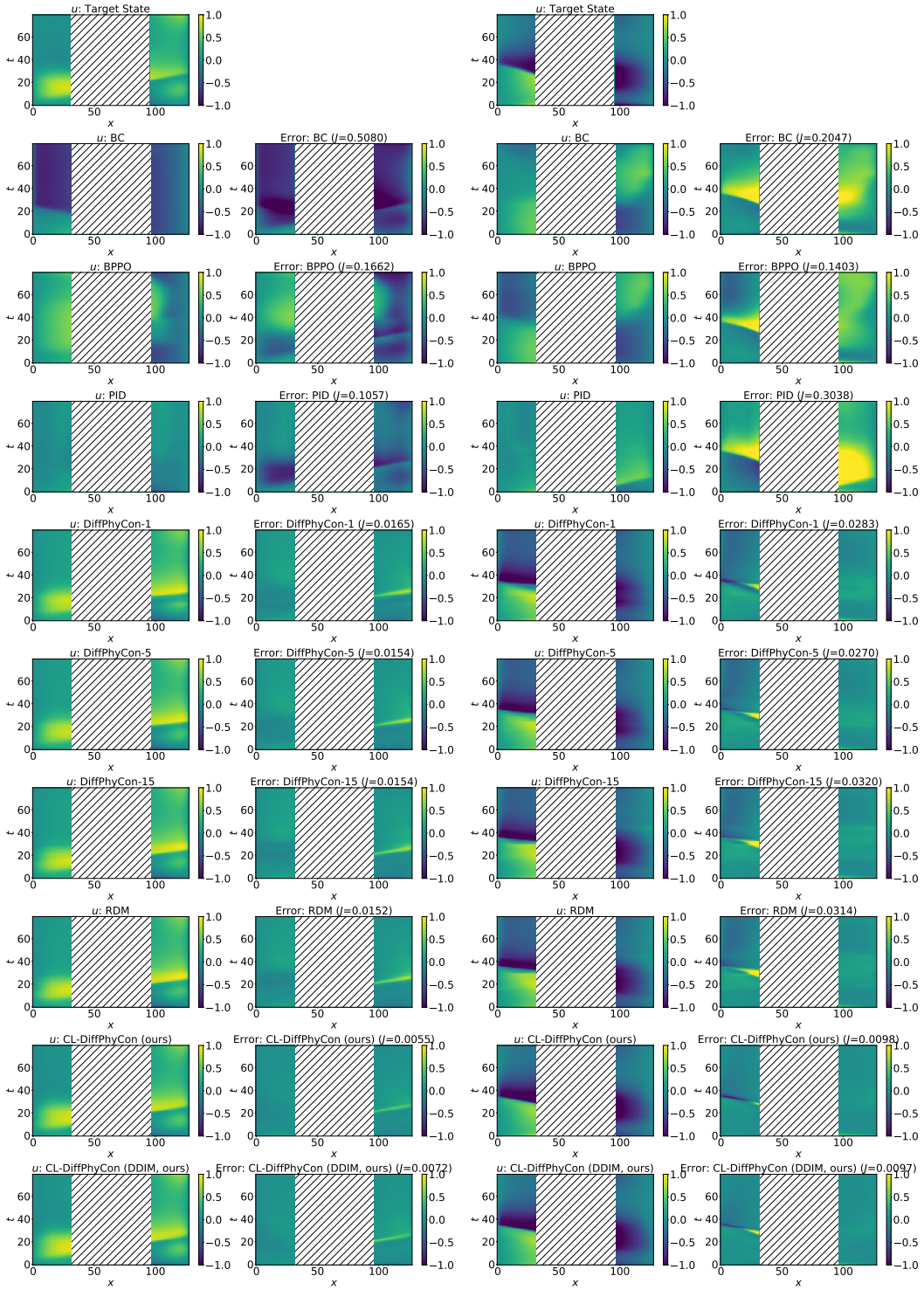

Figure 11: **Two visualizations results of 1D Burgers' equation control under the half domain setting**. The first line is the target $u_d(t, x)$ and the error $u(t, x) - u_d(t, x)$ measures the gap between the state under control and target. The horizontal axis is the time coordinate and the vertical axis is the state. The white area with oblique lines in the middle represents an unobservable area.

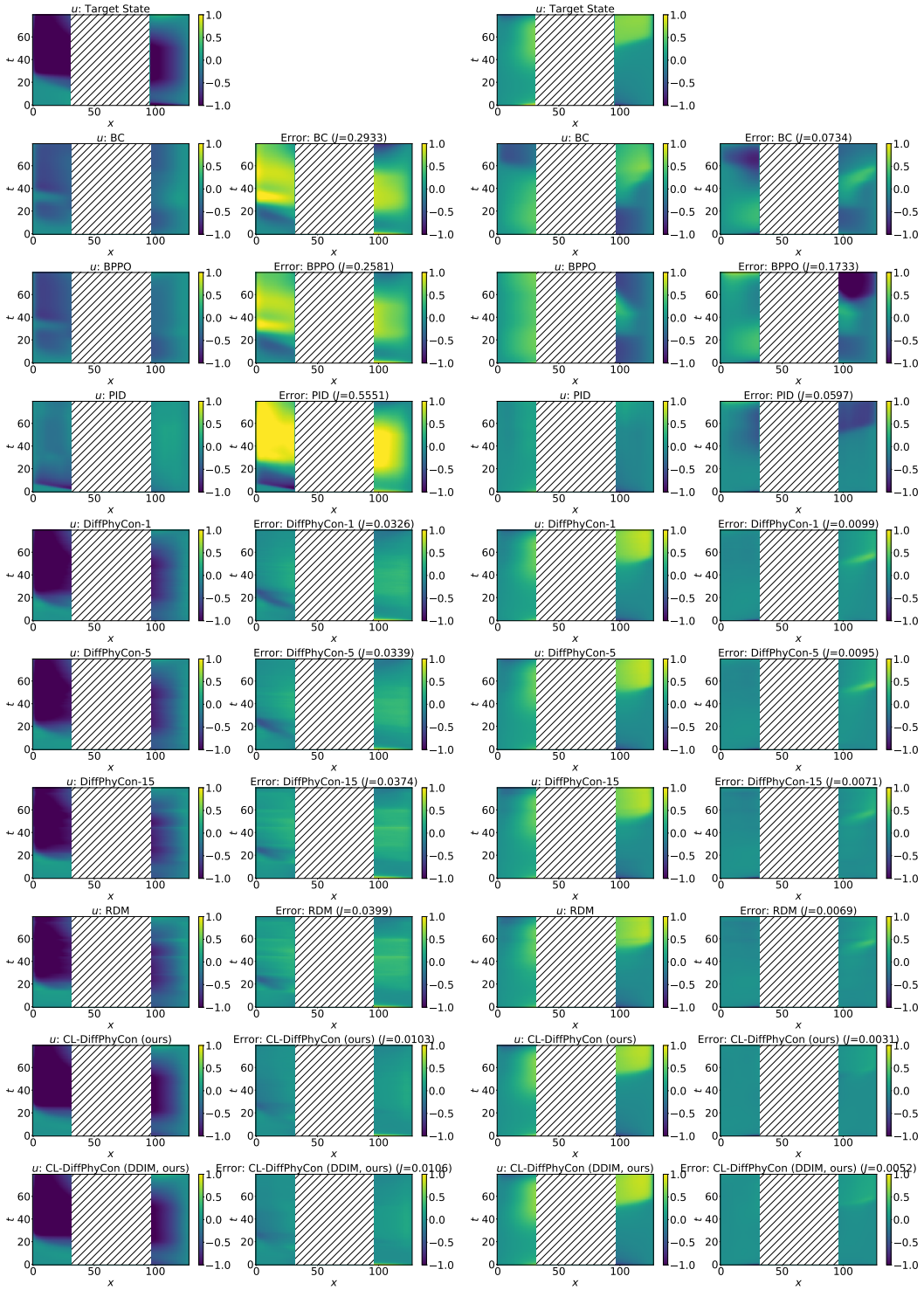

Figure 12: **Two visualizations results of 1D Burgers' equation control under the half domain setting**. The first line is the target $u_d(t, x)$ and the error $u(t, x) - u_d(t, x)$ measures the gap between the state under control and target. The horizontal axis is the time coordinate and the vertical axis is the state. The white area with oblique lines in the middle represents an unobservable area.

Figure 13: **Visualizations results of 2D fluid control by our CL-DiffPhyCon method and baselines for a same test sample**. This is an example of the *fixed map* setting. Each row shows six frames of smoke density. The smaller the value of J, the better the performance.

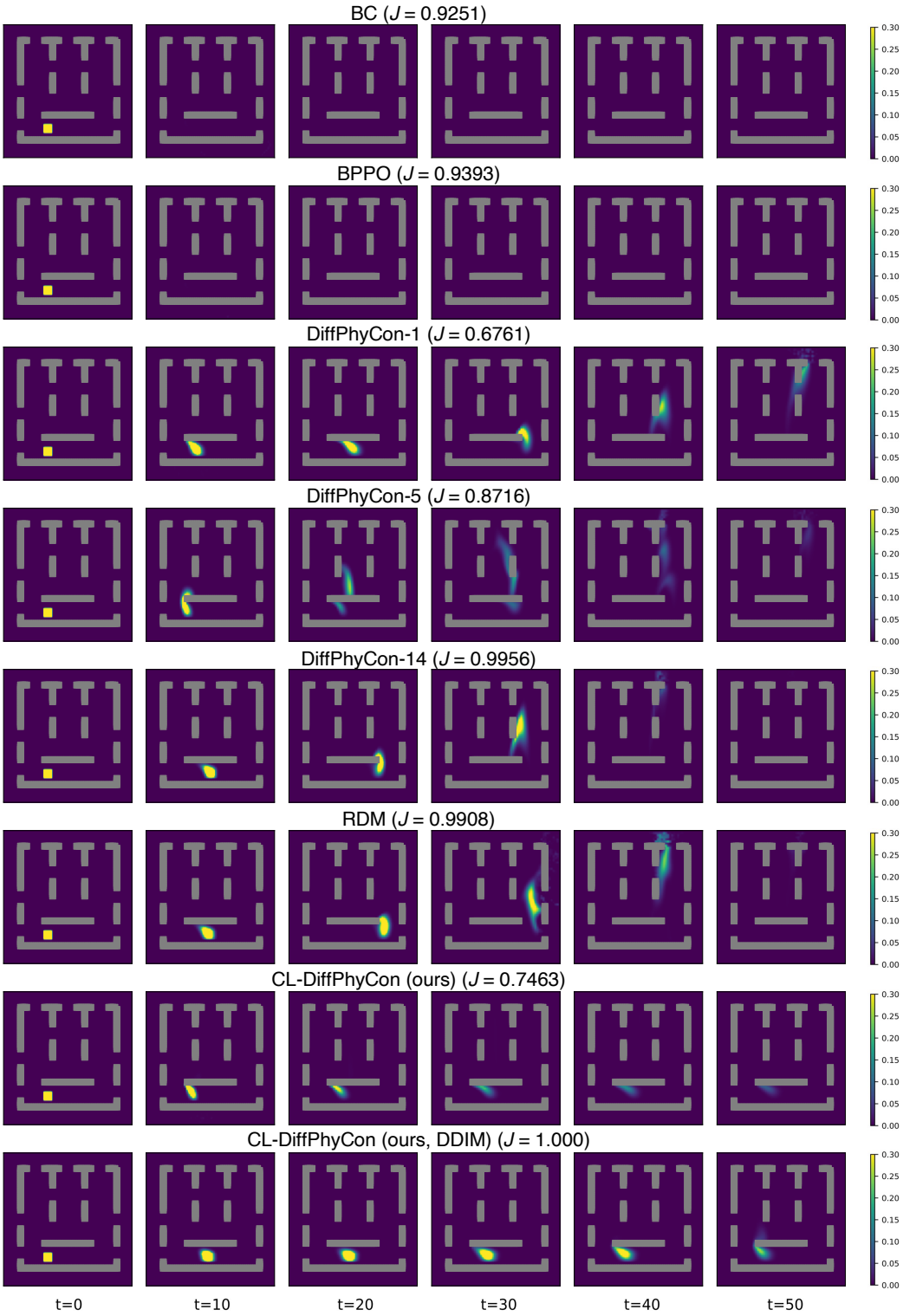

Figure 14: **Visualizations results of 2D fluid control by our CL-DiffPhyCon method and baselines for the same test sample**. This is an example of the *fixed map* setting. Each row shows six frames of smoke density. The smaller the value of J, the better the performance.

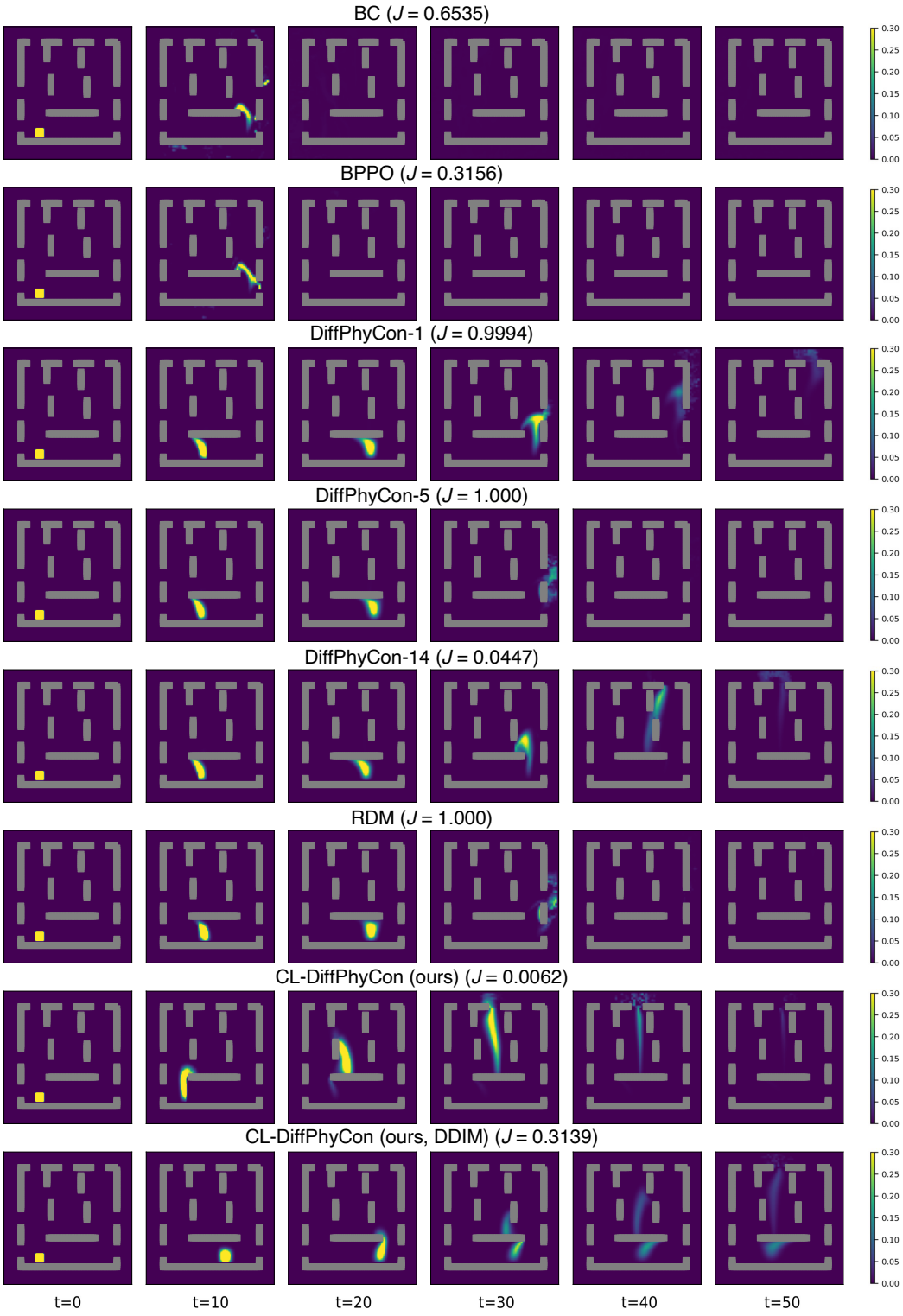

Figure 15: **Visualizations results of 2D fluid control by our CL-DiffPhyCon method and baselines for the same test sample**. This is an example of the *random map* setting. Each row shows six frames of smoke density. The smaller the value of J, the better the performance.

Figure 16: **Visualizations results of 2D fluid control by our CL-DiffPhyCon method and baselines for the same test sample**. This is an example of the *random map* setting. Each row shows six frames of smoke density. The smaller the value of J, the better the performance.