DiffPhyCon: A Generative Approach to Control Complex Physical Systems

Long Wei[1*] **Peiyan Hu**[2*§] **Ruiqi Feng**[1*] **Haodong Feng**[1] **Yixuan Du**[3§] **Tao Zhang**[1]
Rui Wang[4§] **Yue Wang**[5] **Zhi-Ming Ma**[2] **Tailin Wu**[1†]
[1]School of Engineering, Westlake University,
[2]Academy of Mathematics and Systems Science, Chinese Academy of Sciences,
[3]Jilin University, [4]Fudan University, [5]Microsoft AI4Science
`{weilong,hupeiyan,fengruiqi,wutailin}@westlake.edu.cn`

Abstract

Controlling the evolution of complex physical systems is a fundamental task across science and engineering. Classical techniques suffer from limited applicability or huge computational costs. On the other hand, recent deep learning and reinforcement learning-based approaches often struggle to optimize long-term control sequences under the constraints of system dynamics. In this work, we introduce Diffusion Physical systems Control (DiffPhyCon), a new class of method to address the physical systems control problem. DiffPhyCon excels by simultaneously minimizing both the learned generative energy function and the predefined control objectives across the entire trajectory and control sequence. Thus, it can explore globally and plan near-optimal control sequences. Moreover, we enhance DiffPhyCon with prior reweighting, enabling the discovery of control sequences that significantly deviate from the training distribution. We test our method on three tasks: 1D Burgers' equation, 2D jellyfish movement control, and 2D high-dimensional smoke control, where our generated jellyfish dataset is released as a benchmark for complex physical system control research. Our method outperforms widely applied classical approaches and state-of-the-art deep learning and reinforcement learning methods. Notably, DiffPhyCon unveils an intriguing fast-close-slow-open pattern observed in the jellyfish, aligning with established findings in the field of fluid dynamics. The project website, jellyfish dataset, and code can be found at `https://github.com/AI4Science-WestlakeU/diffphycon`.

1 Introduction

Modeling the dynamics of complex physical systems is an important class of problems in science and engineering. Usually, we are not only interested in predicting a physical system's behavior but also injecting time-variant signals to steer its evolution and optimize specific objectives. This gives rise to the complex physical control problem, a fundamental task with wide applications, such as controlled nuclear fusion [10], fluid control [22], underwater devices [70] and aviation [47], among others.

Despite its importance, controlling complex physical systems efficiently presents significant challenges. It inherits the fundamental challenge of simulating complex physical systems, which are typically high-dimensional and highly nonlinear, as specified in Appendix A.1. Furthermore, observed control signals and corresponding system trajectories for optimizing a control model are typically far from the optimal solutions of the specific control objective. This fact poses a significant challenge of dilemma: *How to explore long-term control sequences beyond its training distribution*

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

to seek near-optimal solutions while making the resulting system trajectory faithful to the dynamics of the physical system?

To tackle physical systems control problems, various techniques have been proposed, yet they fall short of addressing the above challenges. Regarding traditional control methods, the Proportional-Integral-Derivative (PID) control [33], is efficient but only suitable for a limited range of problems. Conversely, Model Predictive Control (MPC) [57], despite a wider range of applicability, suffers from high computational costs and challenges in global optimization. Recent advances in supervised learning (SL) [22, 24] and reinforcement learning (RL) [15, 46, 52], trained on system trajectories and control signals data, have demonstrated impressive performance in solving physical systems control problems. However, existing SL and model-based RL methods either fall into myopic failure modes [26] that fail to achieve long-term near-optimal solutions, or produce adversarial state trajectories [72] that violate the physical system's dynamics. The main reason may be that they treat the continuous evolution of dynamics from an iterative view, both lacking long-term vision and struggling in global optimization. See Appendix A.2 for more related work on physical system control.

In this work, we introduce Diffusion Physical systems Control (DiffPhyCon), a *new class* of method to address the physical systems control problem. We take an energy optimization perspective over system trajectory and control sequences across the whole horizon to implicitly capture the constraints inherent in system dynamics. We accomplish this through diffusion models, which are trained using system trajectory data and control sequences. In the inference stage, DiffPhyCon integrates simulation and control optimization into a unified energy optimization process. This prevents the generated system dynamics from falling out of distribution, and offers an enhanced perspective over long-term dynamics, facilitating the discovery of control sequences that optimize the objectives.

An essential aspect of physical systems control lies in its capacity to generate near-optimal controls, even when they may deviate significantly from the training distribution. We address this challenge with the key insight that the learned generative energy landscape can be *decomposed* into two components: a prior distribution representing the control sequence and a conditional distribution characterizing the system trajectories given the control sequence. Based on this insight, we develop a *prior reweighting* technique to subtract the effect of the prior distribution of control sequences, with adjustable strength, from the overall joint generative energy landscape during inference.

In summary, we contribute the following: **(1)** We develop DiffPhyCon, a novel generative method to control complex physical systems. By optimizing trajectory and control sequences jointly in the entire horizon by diffusion models, DiffPhyCon facilitates global optimization of long-term dynamics and helps to reduce myopic failure modes. **(2)** We introduce the prior reweighting technique to plan control sequences that are superior to those in the training set. **(3)** We demonstrate the effectiveness of our method on 1D Burgers' equation, 2D jellyfish movement control, and 2D high-dimensional smoke control tasks. On all three tasks, our method outperforms widely applied classical control methods and is also competitive with recent supervised learning and strong reinforcement learning baselines, particularly demonstrating advantages of DiffPhyCon in scenarios with partial observations and partial/indirect control. Notably, DiffPhyCon reveals the intriguing fast-close-slow-open pattern exhibited by the jellyfish, aligning with findings in the field of fluid dynamics [28]. **(4)** We generate a dataset of 2D jellyfish based on a CFD (computational fluid dynamic) software to mimic the movement of a jellyfish under control signals of its flapping behavior. To advance research in controlling complex physical systems, we release our dataset as a benchmark.

2 Background

2.1 Problem Setup

We consider the following complex physical system control problem:

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \mathcal{J}(\mathbf{u}, \mathbf{w}) \quad \text{s.t.} \quad \mathcal{C}(\mathbf{u}, \mathbf{w}) = 0. \tag{1}$$

Here $\mathbf{u}(t, \mathbf{x}) : [0, \mathcal{T}] \times \Omega \mapsto \mathbb{R}^{d_{\mathbf{u}}}$ is the system trajectory $\{\mathbf{u}(t, \cdot), t \in [0, \mathcal{T}]\}$ defined on time range $[0, \mathcal{T}] \subset \mathbb{R}$ and spatial domain $\Omega \subset \mathbb{R}^D$, and $\mathbf{w}(t, \mathbf{x}) : [0, \mathcal{T}] \times \Omega \mapsto \mathbb{R}^{d_{\mathbf{w}}}$ is the external control signal for the physical system with dimension $d_{\mathbf{w}}$. $\mathcal{J}(\mathbf{u}, \mathbf{w})$ denotes the control objective. For example, $\mathcal{J}$ can be designed to measure the control performance towards a target state $\mathbf{u}^*$ with cost constraints: $\mathcal{J} := \int \|\mathbf{u} - \mathbf{u}^*\|^2 \mathrm{d}\mathbf{x}\mathrm{d}t + \int \|\mathbf{w}\|^2 \mathrm{d}\mathbf{x}\mathrm{d}t$. $\mathcal{C}(\mathbf{u}, \mathbf{w}) = 0$ denotes the physical

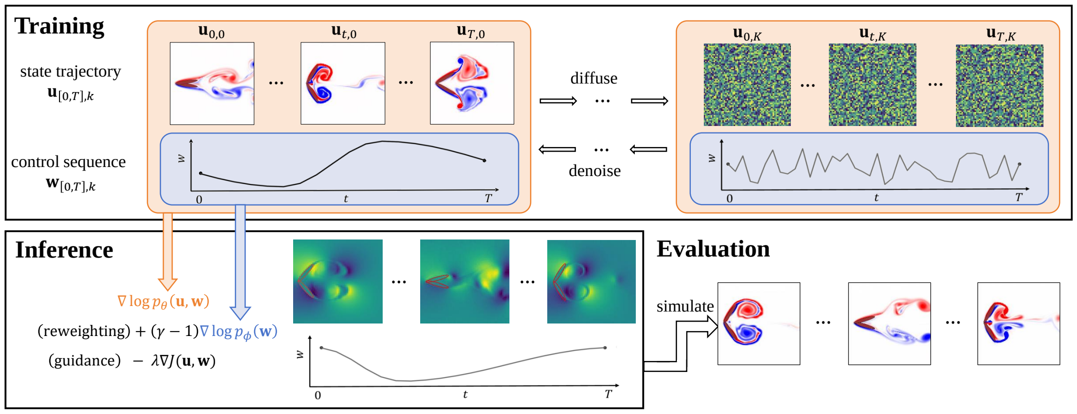
Training
state trajectory $\mathbf{u}_{[0,T],k}$
control sequence $\mathbf{w}_{[0,T],k}$
$\mathbf{u}_{0,0}$
$\mathbf{u}_{t,0}$
$\mathbf{u}_{T,0}$
diffuse
denoise
$\mathbf{u}_{0,K}$
$\mathbf{u}_{t,K}$
$\mathbf{u}_{T,K}$
Inference
$\nabla \log p_\theta(\mathbf{u},\mathbf{w})$
(reweighting) $+ (\gamma-1)\nabla \log p_\phi(\mathbf{w})$
(guidance) $-\lambda \nabla J(\mathbf{u},\mathbf{w})$
Evaluation
simulate

Figure 1: **Overview of DiffPhyCon**. The figure depicts the training (top), inference (bottom left), and evaluation (bottom right) of DiffPhyCon. Orange and blue colors respectively represent models learning the joint distribution $p_\theta(\mathbf{u},\mathbf{w})$ and the prior distribution $p_\phi(\mathbf{w})$. Through prior reweighting and guidance, DiffPhyCon is capable of generating superior control sequences.

constraints. $\mathcal{C}(\mathbf{u},\mathbf{w})=0$ can be specified either *explicitly* by a PDE dynamic which describes how the control signal $\mathbf{w}(t,\mathbf{x})$ drives the trajectory $\mathbf{u}(t,\mathbf{x})$ to evolve under boundary and initial conditions, or *implicitly* by control sequences and trajectory data collected from observation of the physical system. In the latter case, we may even have access to only partial trajectory data or engage in partial control. These situations collectively pose significant challenges to the physical system control task.

2.2 Preliminary: Diffusion Models

Diffusion models [19] are a class of generative models that learn data distribution from data. Diffusion models consist of two opposite processes: the forward process $q(\mathbf{x}_{k+1}|\mathbf{x}_k)=\mathcal{N}(\mathbf{x}_{k+1};\sqrt{\alpha_k}\mathbf{x}_k,(1-\alpha_k)\mathbf{I})$ to corrupt a clean data $\mathbf{x}_0$ to a Gaussian noise $\mathbf{x}_K\sim\mathcal{N}(\mathbf{0},\mathbf{I})$, and the reverse parametrized process $p_\theta(\mathbf{x}_{k-1}|\mathbf{x}_k)=\mathcal{N}(\mathbf{x}_{k-1};\mu_\theta(\mathbf{x}_k,k),\sigma_k\mathbf{I})$ to denoise from standard Gaussian $\mathbf{x}_K\sim\mathcal{N}(\mathbf{0},\mathbf{I})$, where $\{\alpha_k\}_{k=1}^K$ is the variance schedule. To train diffusion models, [19] propose the DDPM method to minimize the following training loss for the denoising network $\boldsymbol{\epsilon}_\theta$, a simplification of the evidence lower bound (ELBO) for the log-likelihood of the data:

$$\mathcal{L}=\mathbb{E}_{k\sim U(1,K),\mathbf{x}_0\sim p(x),\boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}[\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_k}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_k}\boldsymbol{\epsilon},k)\|_2^2], \tag{2}$$

where $\bar{\alpha}_k:=\prod_{i=1}^k\alpha_i$. $\boldsymbol{\epsilon}_\theta$ estimates the noise to be removed to recover data $\mathbf{x}_0$. During inference, iterative application of $\boldsymbol{\epsilon}_\theta$ from a Gaussian noise could generate a new sample $\mathbf{x}_0$ that approximately follows the data distribution $p(\mathbf{x})$. See Appendix A.3 for related work on diffusion models.

Notation. We use $\mathbf{v}_{[n,m]}=[\mathbf{v}_n,\cdots,\mathbf{v}_m]$ to denote a sequence of variables. We use $\mathbf{z}_{[0,T-1],k}$ to denote the hidden variable of $\mathbf{z}_{[0,T-1]}$ in a diffusion step k. For simplicity, we abbreviate $\mathbf{w}_{[0,T-1]}$, $\mathbf{u}_{[1,T]}$ as $\mathbf{w}$, $\mathbf{u}$. Concatenation of two variables is denoted via *e.g.*, $[\mathbf{u},\mathbf{w}]$.

3 Method

In this section, we detail our method DiffPhyCon. In Section 3.1, we introduce our method including its training and inference. In Section 3.2, we further propose a prior reweighting technique to improve DiffPhyCon. The overview of DiffPhyCon is illustrated in Figure 1.

3.1 Generative Control by Diffusion Models

DiffPhyCon takes an energy optimization perspective to solve the problem Eq. (1), where PDE constraints can be modeled as a parameterized energy-based model (EBM) $E_\theta(\mathbf{u},\mathbf{w},\mathbf{c})$ which characterizes the distribution $p(\mathbf{u},\mathbf{w}|\mathbf{c})$ of $\mathbf{u}$ and $\mathbf{w}$ conditioned on conditions $\mathbf{c}$ by the correspondence $p(\mathbf{u},\mathbf{w}|\mathbf{c})\propto\exp(-E_\theta(\mathbf{u},\mathbf{w},\mathbf{c}))$. Lower $E_\theta(\mathbf{u},\mathbf{w},\mathbf{c})$, or equivalently higher $p(\mathbf{u},\mathbf{w}|\mathbf{c})$, means better satisfaction of the PDE constraints. Then the problem Eq. (1) can be converted to:

$$\mathbf{u}^*, \mathbf{w}^* = \operatorname*{argmin}_{\mathbf{u},\mathbf{w}} \left[E_\theta(\mathbf{u}, \mathbf{w}, \mathbf{c}) + \lambda \cdot \mathcal{J}(\mathbf{u}, \mathbf{w}) \right], \tag{3}$$

where λ is a hyperparameter. This formulation optimizes $\mathbf{u}$ and $\mathbf{w}$ of all physical time steps simultaneously. The first term encourages the generated $\mathbf{w}$ and $\mathbf{u}$ to satisfy the PDE constraints $\mathcal{C}(\mathbf{u}, \mathbf{w}) = 0$. The second term guides optimization towards optimal objectives. The advantage of this optimization framework is that it tasks a global optimization on $\mathbf{u}$ and $\mathbf{w}$ of all times steps, which may obtain better solutions that are faithful to the dynamics of the physical system. Details about the effect of the hyperparameter λ are provided in Appendix L.

Training. To train E_θ, we exploit the diffusion model to estimate the gradient of E_θ. For convenience, we introduce a new variable $\mathbf{z}$ to represent the concatenation of $\mathbf{u}$ and $\mathbf{w}$ as $\mathbf{z} = [\mathbf{u}, \mathbf{w}]$. We use a denoising network $\boldsymbol{\epsilon}_\theta$ [19], which approximates $\nabla E_\theta(\mathbf{z}, \mathbf{c})$ [13], to learn the noise that should be denoised in each diffusion step $k = 1, \cdots, K$. The training loss of $\boldsymbol{\epsilon}_\theta$ is:

$$\mathcal{L} = \mathbb{E}_{k \sim U(1,K), (\mathbf{z},\mathbf{c}) \sim p(\mathbf{z},\mathbf{c}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})} [\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_k}\mathbf{z} + \sqrt{1 - \bar{\alpha}_k}\boldsymbol{\epsilon}, \mathbf{c}, k) \|_2^2], \tag{4}$$

where $\boldsymbol{\epsilon}_\theta$ is conditioned on $\mathbf{c}$. Regarding training datasets, they could be generated by designing the conditions $\mathbf{c}$ and control sequences followed by a simulation when an explicit PDE form is available. Otherwise, they should be collected from observed pairs of control sequences and trajectories.

Control optimization. After the denoising network is trained, the Eq. (3) can be optimized by the Langevin sampling procedure as follows. We start from an initial sample $\mathbf{z}_K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and iteratively run the following process

$$\mathbf{z}_{k-1} = \mathbf{z}_k - \eta \left(\nabla_\mathbf{z} (E_\theta(\mathbf{z}_k, \mathbf{c}) + \lambda \mathcal{J}(\hat{\mathbf{z}}_k)) \right) + \xi, \quad \xi \sim \mathcal{N}\big(\mathbf{0}, \sigma_k^2 \mathbf{I}\big), \tag{5}$$

where σ_k^2 and η correspond to noise schedules and scaling factors used in the diffusion process, respectively. Here $\hat{\mathbf{z}}_k$ is the approximate noise-free $\mathbf{z}_0$ estimated from $\mathbf{z}_k$ by:

$$\hat{\mathbf{z}}_k = (\mathbf{z}_k - \sqrt{1 - \bar{\alpha}_k}\boldsymbol{\epsilon}_\theta(\mathbf{z}_k, \mathbf{c}, k)) / \sqrt{\bar{\alpha}_k}. \tag{6}$$

We calculate $\mathcal{J}$ in (5) based on $\hat{\mathbf{z}}_k$ instead of directly using $\mathbf{z}_k$ because otherwise noise in $\mathbf{z}_k$ could bring errors to $\mathcal{J}$. Then ∇E_θ can be replaced by our trained denoising network $\boldsymbol{\epsilon}_\theta$ as follows:

$$\mathbf{z}_{k-1} = \mathbf{z}_k - \eta \left(\boldsymbol{\epsilon}_\theta(\mathbf{z}_k, \mathbf{c}, k) + \lambda \nabla_\mathbf{z} \mathcal{J}(\hat{\mathbf{z}}_k) \right) + \xi, \quad \xi \sim \mathcal{N}\big(\mathbf{0}, \sigma_k^2 \mathbf{I}\big) \tag{7}$$

Iteration of this denoising process for $k = K, K-1, ..., 1$ yields a final solution $\mathbf{z}_0 = \{\mathbf{u}_{[1,T],0}, \mathbf{w}_{[0,T-1],0}\}$ for the optimization problem Eq. (3).

Guidance conditioning. In addition to the above introduced explicit guidance, conditioning is also widely used to guide sampling in diffusion models [20, 58]. When the control objective can be naturally expressed in a conditioning form, e.g., the generated trajectory $\mathbf{u}$ is required to coincide with a desired target $\mathbf{u}^*$, we can include $\mathbf{u} = \mathbf{u}^*$ as a condition in $\mathbf{c}$ such that the sampled trajectory $\mathbf{u}$ from diffusion models automatically satisfying $\mathbf{u} = \mathbf{u}^*$.

Overall, in our proposed DiffPhyCon framework, the control objective $\mathcal{J}$ can be optimized either using the explicit guidance $\nabla \mathcal{J}$ or guidance conditioning depending on the specific control objectives.

3.2 Prior Reweighting

Motivation. In physical systems control, a critical challenge lies in obtaining control sequences superior to those in training datasets, which often deviate significantly from achieving the optimal control objective. Although guidance of the control objective is incorporated in our diffusion model, generated control sequences are still highly influenced by the prior distribution of control sequences in training datasets. This inspires us to explore strategies to mitigate the effect of this prior, aiming to generate near-optimal control sequences.

We address this challenge with the key insight that the energy-based model $E(\mathbf{u}, \mathbf{w}, \mathbf{c})$ can be decomposed into two components: one is $E^{(p)}(\mathbf{w}, \mathbf{c})$ derived from the prior distribution $p(\mathbf{w}|\mathbf{c})$ of control sequences, and the other $E^{(c)}(\mathbf{u}, \mathbf{w}, \mathbf{c})$ representing the conditional probability distribution $p(\mathbf{u}|\mathbf{w}, \mathbf{c})$ of trajectories with respect to given control sequences. This decomposition has the following form

$$E(\mathbf{u}, \mathbf{w}, \mathbf{c}) = E^{(p)}(\mathbf{u}, \mathbf{c}) + E^{(c)}(\mathbf{u}, \mathbf{w}, \mathbf{c}), \tag{8}$$

by the corresponding decomposition of distribution $p(\mathbf{u}, \mathbf{w}|\mathbf{c}) = p(\mathbf{w}|\mathbf{c})p(\mathbf{u}|\mathbf{w}, \mathbf{c})$. We propose a *prior reweighting* technique, which introduces an adjustable hyperparameter $\gamma > 0$ as an exponential of $p(\mathbf{w}|\mathbf{c})$, allowing for the tuning of the influence of this prior distribution. Then we have a reweighted version of $p(\mathbf{u}, \mathbf{w}|\mathbf{c})$ as $p_\gamma(\mathbf{u}, \mathbf{w}|\mathbf{c}) = p(\mathbf{w}|\mathbf{c})^\gamma p(\mathbf{u}|\mathbf{w}, \mathbf{c})/Z$, which is also a probability distribution and can be further transformed to

$$p_\gamma(\mathbf{u}, \mathbf{w}|\mathbf{c}) = p(\mathbf{w}|\mathbf{c})^{\gamma-1} p(\mathbf{u}, \mathbf{w}|\mathbf{c})/Z, \tag{9}$$

where Z is a normalization constant. In particular, when $0 < \gamma < 1$, this approach is advantageous as it flattens the distribution $p(\mathbf{u}, \mathbf{w}|\mathbf{c})$, thereby increasing the likelihood of sampling from low probability region of $p(\mathbf{u}, \mathbf{w}|\mathbf{c})$, where the optimal solutions of the problem Eq. (3) probably lie.

Integrating Eq. (9) into Eq. (8), we have

$$E^{(\gamma)}(\mathbf{u}, \mathbf{w}, \mathbf{c}) = (\gamma - 1)E^{(p)}(\mathbf{w}, \mathbf{c}) + E_\theta(\mathbf{u}, \mathbf{w}, \mathbf{c}) - \log Z, \tag{10}$$

where $E^{(\gamma)}(\mathbf{u}, \mathbf{w}, \mathbf{c}) = -\log\left(p_\gamma(\mathbf{u}, \mathbf{w}|\mathbf{c})\right) + const$ is the reweighted energy-based model associated with $E_\theta(\mathbf{u}, \mathbf{w}, \mathbf{c})$ in Eq. (3), relying on the hyperparameter γ. Then the optimization problem Eq. (3) can be transformed to

$$\mathbf{u}^*, \mathbf{w}^* = \underset{\mathbf{u}, \mathbf{w}}{\operatorname{argmin}} \left[E^{(\gamma)}(\mathbf{u}, \mathbf{w}, \mathbf{c}) + \lambda \cdot \mathcal{J}(\mathbf{u}, \mathbf{w}) \right]. \tag{11}$$

Optimization of this problem encourages sampling from the low likelihood region of $p(\mathbf{w}|\mathbf{c})$ while minimizing the control objective, which possesses the capability to generate control sequences that are more likely to be near-optimal than its degenerate version $\gamma = 1$ in the original optimization problem Eq. (3). The intuition of prior reweighting is illustrated in Figure 2.

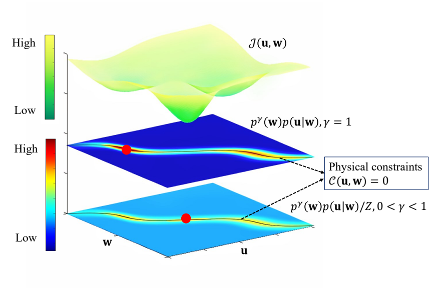
High
Low
$\mathcal{J}(\mathbf{u}, \mathbf{w})$
High
Low
$p^\gamma(\mathbf{w})p(\mathbf{u}|\mathbf{w}), \gamma = 1$
Physical constraints
$\mathcal{C}(\mathbf{u}, \mathbf{w}) = 0$
$p^\gamma(\mathbf{w})p(\mathbf{u}|\mathbf{w})/Z, 0 < \gamma < 1$
w
u

Figure 2: **Intuition of Prior Reweighting**. The top surface illustrates the landscape of $\mathcal{J}(\mathbf{u}, \mathbf{w})$, where the high-dimensional variables $\mathbf{u}$ and $\mathbf{w}$ are represented using one dimension. The middle and lower planes depict probability heatmaps for the reweighted distribution $p^\gamma(\mathbf{w})p(\mathbf{u}|\mathbf{w})/Z$. Adjusting γ from $\gamma = 1$ (middle plane) to $0 < \gamma < 1$ (lower plane), a better minimal of $\mathcal{J}$ (red dot in the lower plane) gains the chance to be sampled. This contrasts with the suboptimal red point in the middle plane highly influenced by the prior $p(\mathbf{w})$.

Training. To learn the reweighted energy $E^{(\gamma)}(\mathbf{u}, \mathbf{w}, \mathbf{c})$, we parameterize its gradient as a summation of two parts by taking the gradient of both sides of Eq. (10):

$$\begin{aligned}\nabla E^{(\gamma)}_{\phi,\theta}(\mathbf{u}, \mathbf{w}, \mathbf{c}) = (\gamma - 1)\nabla E^{(p)}_\phi(\mathbf{w}, \mathbf{c}) \\ + \nabla E_\theta(\mathbf{u}, \mathbf{w}, \mathbf{c}),\end{aligned}$$

where $E^{(p)}_\phi(\mathbf{w}, \mathbf{c})$ parameterizes the energy based model $E^{(p)}(\mathbf{w}, \mathbf{c})$ corresponding to $p(\mathbf{w}|\mathbf{c})$. Note that $\nabla \log Z$ vanishes here because it is a constant. Notice that $\nabla E_\theta(\mathbf{u}, \mathbf{w}, \mathbf{c})$ has already been trained by Eq. (4). $\nabla E^{(p)}_\phi(\mathbf{w}, \mathbf{c})$ can be trained similarly by the following loss function

$$\mathcal{L} = \mathbb{E}_{k \sim U(1,K), (\mathbf{w}, \mathbf{c}) \sim p(\mathbf{w}, \mathbf{c}), \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})}[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\sqrt{\bar{\alpha}_t}\mathbf{w} + \sqrt{1 - \bar{\alpha}_k}\boldsymbol{\epsilon}, \mathbf{c}, k)\|_2^2], \tag{12}$$

where $\boldsymbol{\epsilon}_\phi$ is the conditional denoising network that approximates $\nabla E^{(p)}_\phi(\mathbf{w}, \mathbf{c})$.

Control optimization. With both $\boldsymbol{\epsilon}_\theta$ and $\boldsymbol{\epsilon}_\phi$ trained, Eq. (11) can be optimized by running:

$$\mathbf{z}_{k-1} = \mathbf{z}_k - \eta(\boldsymbol{\epsilon}_\theta(\mathbf{z}_k, \mathbf{c}, k) + \lambda \nabla_{\mathbf{z}} \mathcal{J}(\hat{\mathbf{z}}_k)) + \xi_1, \quad \xi_1 \sim \mathcal{N}\big(0, \sigma_k^2 \mathbf{I}\big) \tag{13}$$

$$\mathbf{w}_{k-1} = \mathbf{w}_{k-1} - \eta(\gamma - 1)\boldsymbol{\epsilon}_\phi(\mathbf{w}_k, \mathbf{c}, k) + \xi_2, \quad \xi_2 \sim \mathcal{N}\big(0, \sigma_k^2 \mathbf{I}\big), \tag{14}$$

iteratively, where $\mathbf{z}_k = [\mathbf{u}_k, \mathbf{w}_k]$. The difference between this iteration scheme and Eq. (7) is that it uses an additional step to update $\mathbf{w}_k$ based on the predicted noise of $\boldsymbol{\epsilon}_\phi$. This guides $\mathbf{z}_k = [\mathbf{u}_k, \mathbf{w}_k]$ to move towards the reweighted distribution $p_\gamma(\mathbf{u}, \mathbf{w}|\mathbf{c})$ while aligning with the direction to decrease the objective by its guidance in the iteration of $\mathbf{z}_k$. The complete algorithm is present in Algorithm 1. Detailed discussion and results about how to set the hyperparameter γ are provided in Appendix L.

Theoretical Analysis. Consider a pair $[\mathbf{u}, \mathbf{w}]$ sampled using the prior reweighting technique with hyperparameter γ: $[\mathbf{u}, \mathbf{w}] \sim p_\gamma(\mathbf{u}, \mathbf{w}) = p(\mathbf{u}|\mathbf{w})p^\gamma(\mathbf{w})/C_\gamma$. Denote $\mathcal{J}^*$ as the global minimum

Algorithm 1 Inference for DiffPhyCon

```
1: Require Diffusion models $\epsilon_\theta(\mathbf{z}_k, \mathbf{c}, k)$ and $\epsilon_\phi(\mathbf{w}_k, \mathbf{c}, k)$, control objective $\mathcal{J}(\cdot)$, covariance matrix $\sigma_k^2 I$,
   control conditions $\mathbf{c}$, schedule $\bar{\alpha}_k$, hyperparameters $\lambda, \gamma, K$
2: Initialize optimization variables $\mathbf{z}_K = [\mathbf{u}_K, \mathbf{w}_K] \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: for $k = K, \dots, 1$ do
4:     $\hat{\mathbf{z}}_k = (\mathbf{z}_k - \sqrt{1-\bar{\alpha}_k}\epsilon_\theta(\mathbf{z}_k, \mathbf{c}, k))/\sqrt{\bar{\alpha}_k}$
5:     $\mathbf{z}_{k-1} = \mathbf{z}_k - \eta(\epsilon_\theta(\mathbf{z}_k, \mathbf{c}, k) + \lambda\nabla_\mathbf{z}\mathcal{J}(\hat{\mathbf{z}}_k)) + \xi_1, \xi_1 \sim \mathcal{N}(0, \sigma_k^2\mathbf{I})$ // transition to next diffusion step
6:     $\mathbf{w}_{k-1} = \mathbf{w}_{k-1} - \eta(\gamma - 1)\epsilon_\phi(\mathbf{w}_k, \mathbf{c}, k) + \xi_2, \xi_2 \sim \mathcal{N}(0, \sigma_k^2\mathbf{I})$ // prior reweighting
7: end for
8: return $\mathbf{u}^*, \mathbf{w}^* = \mathbf{z}_0$
```

of the control objective $\mathcal{J}$. Define $Q(\varepsilon)$ to be the "ϵ-optimal" solution set of $[\mathbf{u}, \mathbf{w}]$ such that $\mathcal{J}(\mathbf{u}, \mathbf{w}) - \mathcal{J}^* \leq (\epsilon)$, whose complement set is $Q(\varepsilon)^c$, and denote $\mathbb{I}_{Q(\varepsilon)}(\mathbf{u}, \mathbf{w})$ as its indicator function, i.e. $\mathbb{I}_{Q(\varepsilon)}(\mathbf{u}, \mathbf{w}) = 1$ if $[\mathbf{u}, \mathbf{w}] \in Q(\varepsilon)$; otherwise 0. Define Y to be the random variable of "whether use $\mathcal{J}$ as a guidance for sampling", namely,

$$p(Y|\mathbf{u}, \mathbf{w}) = \begin{cases} e^{-\mathcal{J}(\mathbf{u},\mathbf{w})}/Z, & Y = 1 \\ \\ 1 - e^{-\mathcal{J}(\mathbf{u},\mathbf{w})}/Z, & Y = 0. \end{cases} \tag{15}$$

Consider $E(\gamma) = \mathbb{E}_{(\mathbf{u},\mathbf{w})\sim p_\gamma(\mathbf{u},\mathbf{w})}[\mathbb{I}_{Q(\varepsilon)}(\mathbf{u}, \mathbf{w})|Y = 1]$, which indicates the expectation of getting an ϵ-optimal solution by using the prior reweighting technique with γ under the guidance of $\mathcal{J}$. Define

$$F(\gamma) = \frac{\mathbb{E}_{\mathbf{u},\mathbf{w}}[\mathbb{I}_{Q(\varepsilon)}(\mathbf{u}, \mathbf{w})p(Y = 1|\mathbf{u}, \mathbf{w}) \ln(p(\mathbf{w}))]}{\mathbb{E}_{\mathbf{u},\mathbf{w}}[\mathbb{I}_{Q(\varepsilon)}(\mathbf{u}, \mathbf{w})p(Y = 1|\mathbf{u}, \mathbf{w})]} - \frac{\mathbb{E}_{\mathbf{u},\mathbf{w}}[\mathbb{I}_{Q(\varepsilon)^c}(\mathbf{u}, \mathbf{w})p(Y = 1|\mathbf{u}, \mathbf{w}) \ln(p(\mathbf{w}))]}{\mathbb{E}_{\mathbf{u},\mathbf{w}}[\mathbb{I}_{Q(\varepsilon)^c}(\mathbf{u}, \mathbf{w})p(Y = 1|\mathbf{u}, \mathbf{w})]},$$

then we have the following theorem (please refer to Appendix B for its proof):

Theorem 3.1. *Assume $E(\gamma)$ is a smooth function, then the following hold:*

- *If $F(1) < 0$, there exists a $\gamma_- < 1$, s.t., $E(\gamma_-) > E(1)$;*
- *If $F(1) > 0$, there exists a $\gamma_+ > 1$, s.t., $E(\gamma_+) > E(1)$.*

Remark: Here $F(\gamma)$ can be interpreted as some kind of difference between "entropies" in $Q(\varepsilon)^c$ and $Q(\varepsilon)$. When $F(1) < 0$, it means that $Q(\varepsilon)^c$ has higher "entropies", implying that the training trajectories are far from optimal. As a result, we may need to flatten the distribution of training trajectories, which corresponds to using the prior reweighting technique with $\gamma < 1$. Since this is the most common case in real scenarios, we usually set $\gamma < 1$.

DiffPhyCon-lite. The introduction of the prior reweighting technique in DiffPhyCon involves training and evaluation of two models, thus bringing in additional computational cost. It is gratifying to note that we can balance the control performance and computational overhead of DiffPhyCon by adjusting the parameter γ. When $\gamma = 1$, the model ϵ_ϕ is not needed, and we denote this simplified version of DiffPhyCon as DiffPhyCon-lite.

4 Experiments

In this section, we aim to answer the following questions: (1) Can DiffPhyCon present superiority over traditional, supervised learning, and reinforcement learning methods for physical systems control? (2) Does the proposed prior reweighting technique help achieve better control objectives? (3) Could answers to (1) and (2) be generalized to more challenging partial observation or partial control scenarios? To answer these questions, we conduct experiments on three vital and challenging problems: 1D Burgers' equation, 2D jellyfish movement control, and 2D smoke control problems.

The following state-of-the-art control methods are selected as baselines. For the 1D Burgers' equation, we use (1) the classical and widely used control algorithm Proportional-Integral-Derivative (PID) [33] interacting with our trained surrogate model of the solver; (2) Supervised Learning method (SL) [24]; RL methods including (3) Soft Actor-Critic (SAC) [16] with offline and surrogate-solver versions; (4) Behaviour Cloning (BC) [49]; and (5) Behavior Proximal Policy Optimization (BPPO)

Table 1: **Best $\mathcal{J}_{\text{actual}}$ achieved in 1D Burgers's equation control.** Bold font denotes the best model, and underline denotes the second best model.

	PO-FC	FO-PC	PO-PC
PID (surrogate-solver)	-	0.09115	0.09631
SL	0.09752	<u>0.00078</u>	0.02328
SAC (surrogate-solver)	0.01577	0.03426	0.02149
SAC (offline)	0.03201	0.04333	0.03328
BC	0.02836	0.00856	0.00952
BPPO	0.02771	0.00852	<u>0.00891</u>
DiffPhyCon-lite (ours)	<u>0.01139</u>	**0.00037**	**0.00494**
DiffPhyCon (ours)	**0.01103**	**0.00037**	**0.00494**

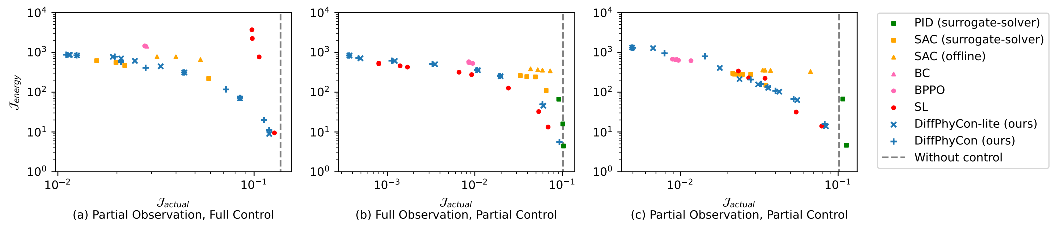

Figure 3: **Pareto frontier of $\mathcal{J}_{\text{energy}}$ vs. $\mathcal{J}_{\text{actual}}$ of different methods for 1D Burgers' equation.**

[73]. Specifically, the surrogate-solver version of SAC interacts with our trained surrogate model of the solver, while the offline version only uses given data. BC and BPPO are also in offline versions. For 2D jellyfish movement control, baselines include SL, SAC (offline), SAC (surrogate-solver), BC, BPPO, and an additional classical multi-input multi-output algorithm Model Predictive Control (MPC) [57]. PID is inapplicable to this data-driven task [2]. Detailed descriptions of baselines are provided in Appendix H and Appendix I.

4.1 1D Burgers' Equation Control

Experiment settings. The Burgers' equation is a governing law occurring in various physical systems. We consider the 1D Burgers' equation with the Dirichlet boundary condition and external force $w(t,x)$, which is also studied in [24, 42].

$$\begin{cases} \frac{\partial u}{\partial t} = -u \cdot \frac{\partial u}{\partial x} + \nu \frac{\partial^2 u}{\partial x^2} + w(t,x) & \text{in } [0,T] \times \Omega \\ u(t,x) = 0 & \text{on } [0,T] \times \partial\Omega \\ u(0,x) = u_0(x) & \text{in } \{t=0\} \times \Omega. \end{cases} \tag{16}$$

Here ν is the viscosity parameter, and $u_0(\mathbf{x})$ is the initial condition. Subject to Eq. (16), given a target state $u_d(x)$, the objective of control is to minimize the control error $\mathcal{J}_{\text{actual}}$ between u_T and u_d, while constraining the energy cost $\mathcal{J}_{\text{energy}}$ of the control sequence $w(t,x)$:

$$\mathcal{J}_{\text{actual}} := \int_{\Omega} |u(T,x) - u_d(x)|^2 \mathrm{d}x, \quad \mathcal{J}_{\text{energy}} := \int_{[0,T]\times\Omega} |w(t,x)|^2 \mathrm{d}t\mathrm{d}x. \tag{17}$$

To make the evaluation challenging, we select three different experiment settings that correspond to different real-life scenarios: partial observation, full control (PO-FC), full observation, partial control (FO-PC), and partial observation, partial control (PO-PC), which are elaborated on in Appendix D.2 and illustrated in Appendix C.1. These settings are challenging for classical control methods such as PID since they require capturing the long-range dependencies in the system dynamics. Note that the reported metrics in different settings are not directly comparable. In this experiment, DiffPhyCon uses the guidance conditioning (Section 3.1) to optimize $\mathcal{J}_{\text{actual}}$ and explicit guidance to optimize the energy cost, which is elaborated in Appendix D.3.

Results. In Table 1, we report results of the control error $\mathcal{J}_{\text{actual}}$ of different methods. It can be observed that DiffPhyCon delivers the best results compared to all baselines. Specifically, DiffPhyCon decreases $\mathcal{J}_{\text{actual}}$ of the best baseline by 30.1%, 52.6%, and 44.6% in the PO-FC, FO-PC, and PO-PC

Table 2: **Jellyfish movement dataset outline.**

training trajectories	test trajectories	resolution	#fluid features	trajectory length
30000	1000	128×128	3	40

settings respectively. From Table 1, DiffPhyCon and DiffPhyCon-lite show little performance gap. This is because, the prior distribution of w that our DiffPhyCon-lite learned is conditioned on both u_0 and u_T, which fully determines the optimal w. Therefore, $p(w|u_0, u_T)$ is intrinsically the optimal distribution and thus DiffPhyCon-lite can already deliver satisfactory performance.

To compare the ability of different methods to optimize $\mathcal{J}_{\text{actual}}$ with constrained energy cost $\mathcal{J}_{\text{energy}}$, we compare the Pareto frontiers of different methods in Figure 3. We vary the hyperparameter λ to control the tradeoff between $\mathcal{J}_{\text{actual}}$ and the energy cost since most baselines have this hyperparameter. As can be observed in Figure 3, the Pareto frontiers of DiffPhyCon are consistently among the best, achieving the *lowest* $\mathcal{J}_{\text{actual}}$ for most settings of the energy budget. Although SL performs well in full observation setting (b) where the system dynamics can be more easily predicted, it encounters difficulty in partial observation scenarios (a)(c). The results demonstrate DiffPhyCon's ability to generate near-optimal control sequences compared to baselines. More visualization results are provided in Appendix C.1. More results of evaluation are presented in Appendix K.1. For efficiency evaluation of training and test phases, please refer to Table 23 in Appendix K.4.

4.2 2D Jellyfish Movement Control

Experiment settings. This task is to control the movement of a flapping jellyfish with two wings in a 2D fluid field where fluid flows at a constant speed. The jellyfish is propelled by the fluid when its wings flap. Its moving speed and efficiency are determined by the mode of flapping. This task is an important source of inspiration for the design of underwater and aerial devices, and its challenges come from complex vortice behavior and fluid-solid coupling dynamics [54, 28]. For this task, fluid dynamics follows the 2D incompressible Navier-Stokes Equation:

$$\begin{cases} \frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v} - \nu\nabla^2\mathbf{v} + \nabla p = 0 \\ \nabla\cdot\mathbf{v} = 0 \\ \mathbf{v}(0,\mathbf{x}) = \mathbf{v}_0(\mathbf{x}), \end{cases} \tag{18}$$

where $\mathbf{v}$ represents the 2D velocity of the fluid, and p represents the pressure, constituting the PDE state $\mathbf{u} = (\mathbf{v}, p)$. The initial velocity condition is $\mathbf{v}_0(\mathbf{x})$ and the kinematic viscosity is ν. We assume that each wing is rigid, so the jellyfish's boundary can be parameterized by the opening angle $\mathbf{w}_t$ of wings. Long-term movement of jellyfish usually presents a periodic flapping mode. Consequently, the control objective is to maximize its average moving speed $\bar{v}$ determined by the pressure of the fluid, under the energy cost constraint $R(\mathbf{w})$ and the periodic constraint $d(\mathbf{w}_T, \mathbf{w}_0)$ of the movement:

$$\mathcal{J} = -\bar{v} + \zeta\cdot R(\mathbf{w}) + d(\mathbf{w}_T, \mathbf{w}_0), \tag{19}$$

subject to Eq. (18) and the boundary condition that the velocity of fluid vanishes near the boundary. The hyperparameter ζ is set to be 1000. We evaluate in two settings: full observation, where the full state $\mathbf{u} = (\mathbf{v}, p)$ is observed; and partial observation, where only pressure is observed. This task is very challenging due to the complicated dynamics of fluid-solid interactions and vortices behaviour [54, 28], especially in the scenario of partial observation where the missing of $\mathbf{v}$ in Eq. (18) restricts the information available to generate well-informed control signals. Details of the experiment are provided in Appendix F.

Open-source Dataset Description. We use the Lily-Pad simulator [66] to generate a dataset describing the movement of jellyfish under the control of its flapping behavior. Statistics about the dataset are listed in Table 2. The feature of this dataset is that it contains a rich variation of vortices behavior and fluid-solid interaction dynamics, determined by violent changes in opening angles of wings and open-close phase ratio. It would serve as an important benchmark for studying complex physical system control problems. Details about the dataset are provided in Appendix E.

Results. Evaluation results are presented in Table 3. It can be seen that our method outperforms the baselines by a large margin in optimizing the control objective $\mathcal{J}$. At a cost of slightly increasing the control cost $R(\mathbf{w})$, control sequences generated by our method achieve a much faster average speed

Table 3: **2D jellyfish movement control results.** Bold font denotes the best model, and underline denotes the second best model.

	Full observation			Partial observation		
	$\bar{v} \uparrow$	$R(\mathbf{w}) \downarrow$	$\mathcal{J} \downarrow$	$\bar{v} \uparrow$	$R(\mathbf{w}) \downarrow$	$\mathcal{J} \downarrow$
MPC	25.72	0.0112	109.17	-150.51	0.1791	329.59
SL	-76.94	0.1286	205.57	-102.98	0.1188	221.79
SAC (surrogate-solver)	-166.96	0.0069	18.14	-153.09	0.0057	158.82
SAC (offline)	-158.66	0.0069	165.58	-206.21	0.0058	211.96
BC	30.48	0.0629	32.44	20.08	0.0556	35.48
BPPO	<u>107.67</u>	0.0867	<u>-20.93</u>	<u>54.83</u>	0.0518	<u>-3.02</u>
DiffPhyCon-lite (ours)	95.04	0.0746	-20.47	2.92	0.0779	74.97
DiffPhyCon (ours)	**279.87**	0.2058	**-74.11**	**150.21**	0.1269	**-23.32**

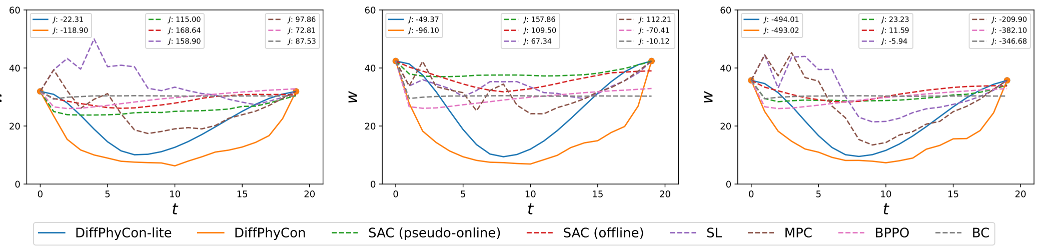

Figure 4: **Comparison of generated control curves of three test jellyfish.** The resulting control objective $\mathcal{J}$ for each curve is presented.

than baselines. In the full observation setting, the control objective achieved by DiffPhyCon is -53.18 lower than the best baseline BPPO, while the average speed exhibits an increment of 172.2 over it. This demonstrates that diffusion models are effective for this challenging control task by performing global optimization of trajectory and control sequences in a generative approach. Comparison between DiffPhyCon-lite and DiffPhyCon reveals that by flattening the prior distribution of control sequences, another significant improvement is further achieved. Even in the more challenging partial observation setting, DiffPhyCon still exhibits substantial advantages over existing methods. This reflects our method has a strong control capability under inadequate information. Configuration of the hyperparameter γ in DiffPhyCon and performance with respect to varying γ is presented in Figure 17 in Appendix L.1. Details about the hyperparameter λ can be found in Table 33 in Appendix L.2.

Figure 4 visualizes generated opening angle curves of different methods on three test jellyfish. Opening angle curves of DiffPhyCon-lite show an obvious fast-close-slow-open shape, which is proven to produce high speed in jellyfish movement [28]. The reason is that the fast closing of wings leads to a thrust from fluid in the early state, resulting in a long-term high speed, and followed by a slow opening to reduce resistance. While this mode of movement appears rarely in the training dataset, DiffPhyCon-lite could generate such control sequences for most test samples. This reflects that diffusion models under guidance are effective in optimizing the control objective. Furthermore, DiffPhyCon makes this mode of movement more aggressive, with sharper change of the opening angle in the beginning and end stage within a period. This provides strong evidence that reweighting the prior distribution of control sequences allows for flexible sampling over a flattened distribution, thus control sequences with low prior but good objectives are more likely to be sampled. The movement and the resulting fluid field of the jellyfish corresponding to the middle subfigure of Figure 4 controlled by DiffPhyCon is illustrated in Figure 5 and more examples are provided in Figure 10 in Appendix C.2. Conversely, opening angles obtained by baselines are inferior. The reason may be that they predict opening angles sequentially and hard to globally optimize the objective with three conflicting terms: average speed, $R(\mathbf{w})$, and $d(\mathbf{w}_T, \mathbf{w}_0)$. We further study such myopic failure mode of SAC in Appendix K.3. More comparisons about the variation of the weight ζ are listed in Table 20 and Table 21 in Appendix K.2. For efficiency evaluation of training and test phases, please refer to Table 24 in Appendix K.4. We also extend this experiment to a high-dimensional control signal setting, where the wings of the jellyfish are assumed to be soft. We find that our method is still competitive with baselines. Details about this evaluation are provided in Appendix M.

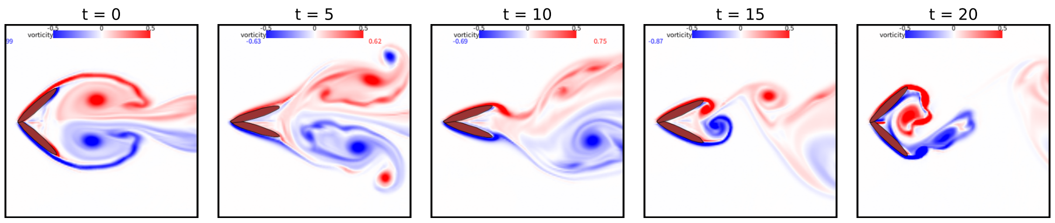

Figure 5: **Visualization of jellyfish movement and fluid field controlled by DiffPhyCon as in the middle subfigure of Figure 4.**

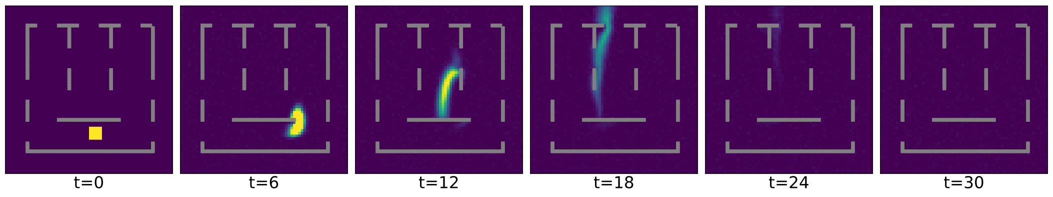

Figure 6: **Visualization of smoke density and fluid field dynamics controlled by DiffPhyCon.**

4.3 2D Smoke Indirect Control

Experiment settings. This task is to control smoke control in an incompressible fluid environment, following a similar (but more challenging) setup of [22]. Control forces were applied within a 64×64 grid flow field, excluding a semi-enclosed region, to minimize the smoke failing to pass through the top middle exit (seven exits in total). For illustration of our settings, please refer to Figure 14 in Appendix G. This high-dimensional indirect control problem involves managing 2D forces at approximately 1,700 grid points every time step, resulting in about 100,000 control variables across 32 time steps, making it highly challenging.

Results. Evaluation results are presented in Table 4. Our method still has significant advantages over baselines in minimizing the control objective. Furthermore, the prior reweighting technique achieves extra improvement over DiffPhyCon-lite. One test sample of smoke density and fluid field dynamics is illustrated in Figure 6. More visualization results of our method are presented in Figure 11 in Appendix G. These results demonstrate that DiffPhyCon is capable of controlling high dimensional physical systems even when control signals are indirectly applied to the system.

Table 4: **2D smoke movement control results.** Bold font denotes the best model, and underline denotes the second best model.

Method	$\mathcal{J} \downarrow$
BC	0.3085
BPPO	0.3066
SAC (surrogate-solver)	0.3212
SAC(offline)	0.6503
DiffPhyCon-lite (ours)	<u>0.2324</u>
DiffPhyCon (ours)	**0.2254**

5 Conclusion

In this work, we have introduced DiffPhyCon, a novel methodology for controlling complex physical systems. It generates control sequences and state trajectories by jointly optimizing the generative energy and control objective. We further introduced prior reweighting to enable the discovery of control sequences that diverge significantly from training. Through comprehensive experiments, we demonstrated our method's superior performance compared to classical, deep learning, and reinforcement learning baselines in challenging physical systems control tasks. We discuss limitation and future work in Appendix N and state social impact in O.

6 Acknowledgment

We thank Yuchen Yang for insightful discussions on theoretical analysis. We thank the anonymous reviewers for providing valuable feedback on our manuscript. We also 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

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[2] 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.

[3] Fan Bao, Min Zhao, Zhongkai Hao, Peiyao Li, Chongxuan Li, and Jun Zhu. Equivariant energy-guided SDE for inverse molecular design. In *The Eleventh International Conference on Learning Representations*, 2023.

[4] 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.

[5] Johannes Brandstetter, Rianne van den Berg, Max Welling, and Jayesh K Gupta. Clifford neural layers for PDE modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[6] Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[7] Salva Rühling Cachay, Bo Zhao, Hailey James, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *arXiv preprint arXiv:2306.01984*, 2023.

[8] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *CoRR*, abs/2105.09506, 2021.

[9] 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.

[10] 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.

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

[12] Jie Ding, Min Wu, and Min Xiao. Nonlinear decoupling control with pi $^{\lambda}$ d $^{\mu}$ neural network for mimo systems. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–8, 2022.

[13] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International Conference on Machine Learning*, pages 8489–8510. PMLR, 2023.

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

[15] Amir-massoud Farahmand, Saleh Nabi, and Daniel N. Nikovski. Deep reinforcement learning for partial differential equation control. In *2017 American Control Conference (ACC)*, pages 3120–3127, 2017.

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

[17] 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.

[18] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. *Advances in neural information processing systems*, 36, 2024.

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

[20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[21] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv:2204.03458*, 2022.

[22] Philipp Holl, Nils Thuerey, and Vladlen Koltun. Learning to control pdes with differentiable physics. In *International Conference on Learning Representations*, 2020.

[23] Benjamin Holzschuh, Simona Vegetti, and Nils Thuerey. Solving inverse physics problems with score matching. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[24] Rakhoon Hwang, Jae Yong Lee, Jin Young Shin, and Hyung Ju Hwang. Solving pde-constrained control problems using operator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4504–4512, 2022.

[25] Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. *arXiv preprint arXiv:2212.11972*, 2022.

[26] 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, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 9902–9915. PMLR, 17–23 Jul 2022.

[27] 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 2022.

[28] Linlin Kang, An-Kang Gao, Fei Han, Weicheng Cui, and Xi-Yun Lu. Propulsive performance and vortex dynamics of jellyfish-like propulsion with burst-and-coast strategy. *Physics of Fluids*, 35(9), 2023.

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, et al. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023.

[31] Leon Lapidus and George F Pinder. *Numerical solution of partial differential equations in science and engineering*. John Wiley & Sons, 1999.

[32] A Larcher and E Hachem. A review on deep reinforcement learning for fluid mechanics: An update. *Physics of Fluids*, 34(11), 2022.

[33] 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.

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

[35] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[36] Jacques Louis Lions. *Optimal control of systems governed by partial differential equations*, volume 170. Springer, 1971.

[37] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[38] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B Tenenbaum. Compositional visual generation with composable diffusion models. In *European Conference on Computer Vision*, pages 423–439. Springer, 2022.

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

[40] Rajat Mittal and Gianluca Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[41] Keith W Morton and David Francis Mayers. *Numerical solution of partial differential equations: an introduction*. Cambridge university press, 2005.

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

[43] Alex Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. *ArXiv*, February 2021.

[44] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. *Proceedings of Machine Learning Research*, 162:16784–16804, 17–23 Jul 2022.

[45] 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.

[46] Yangchen Pan, Amir-massoud Farahmand, Martha White, Saleh Nabi, Piyush Grover, and Daniel Nikovski. Reinforcement learning with function-valued action spaces for partial differential equation control. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3986–3995. PMLR, 10–15 Jul 2018.

[47] Aditya A Paranjape, Jinyu Guan, Soon-Jo Chung, and Miroslav Krstic. Pde boundary control for flexible articulated wings on a robotic aircraft. *IEEE Transactions on Robotics*, 29(3):625–640, 2013.

[48] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

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

[50] 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.

[51] Bartosz Protas. Adjoint-based optimization of pde systems with alternative gradients. *Journal of Computational Physics*, 227(13):6490–6510, 2008.

[52] 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.

[53] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[54] Leif Ristroph and Stephen Childress. Stable hovering of a jellyfish-like flying machine. *Journal of The Royal Society Interface*, 11(92):20130992, 2014.

[55] 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*, pages 234–241. Springer, 2015.

[56] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International conference on machine learning*, pages 8459–8468. PMLR, 2020.

[57] 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.

[58] Dule Shu, Zijie Li, and Amir Barati Farimani. A physics-informed diffusion model for high-fidelity flow field reconstruction. *Journal of Computational Physics*, 478:111972, 2023.

[59] Sabrine Slama, Ayachi Errachdi, and Mohamed Benrejeb. Neural adaptive pid and neural indirect adaptive control switch controller for nonlinear mimo systems. *Mathematical Problems in Engineering*, 2019, 2019.

[60] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[61] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in Neural Information Processing Systems*, 33:6111–6122, 2020.

[62] 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.

[63] Pantelis R Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[64] Marin Vlastelica, Tatiana Lopez-Guevara, Kelsey R Allen, Peter Battaglia, Arnaud Doucet, and Kim Stachenfeld. Diffusion generative inverse design. 2023.

[65] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[66] Gabriel D Weymouth. Lily pad: Towards real-time interactive computational fluid dynamics. *arXiv preprint arXiv:1510.06886*, 2015.

[67] Tailin Wu, Takashi Maruyama, and Jure Leskovec. Learning to accelerate partial differential equations via latent global evolution. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, LA, USA, November 28 - December 9, 2022*, 2022.

[68] Tailin Wu, Takashi Maruyama, Long Wei, Tao Zhang, Yilun Du, Gianluca Iaccarino, and Jure Leskovec. Compositional generative inverse design. *arXiv preprint arXiv:2401.13171*, 2024.

[69] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[70] Shuang Zhang, Xinyu Qian, Zhijie Liu, Qing Li, and Guang Li. Pde modeling and tracking control for the flexible tail of an autonomous robotic fish. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(12):7618–7627, 2022.

[71] Min Zhao, Fan Bao, Chongxuan Li, and Jun Zhu. Egsde: Unpaired image-to-image translation via energy-guided stochastic differential equations. *Advances in Neural Information Processing Systems*, 35:3609–3623, 2022.

[72] Qingqing Zhao, David B Lindell, and Gordon Wetzstein. Learning to solve PDE-constrained inverse problems with graph networks. *Proceedings of Machine Learning Research*, 162:26895–26910, 17–23 Jul 2022.

[73] Zifeng Zhuang, Kun Lei, Jinxin Liu, Donglin Wang, and Yilang Guo. Behavior proximal policy optimization. *arXiv preprint arXiv:2302.11312*, 2023.