PIRF: Physics-Informed Reward Fine-Tuning for Diffusion Models

Mingze Yuan
Harvard University
mingzeyuan@g.harvard.edu

Pengfei Jin
Massachusetts General Hospital
pjin1@mgh.harvard.edu

Na Li*
Harvard University
nali@seas.harvard.edu

Quanzheng Li*
Massachusetts General Hospital
li.quanzheng@mgh.harvard.edu

Abstract

Diffusion models have demonstrated strong generative capabilities across scientific domains, but often produce outputs that violate physical laws. We propose a new perspective by framing physics-informed generation as a sparse reward optimization problem, where adherence to physical constraints is treated as a reward signal. This formulation unifies prior approaches under a reward-based paradigm and reveals a shared bottleneck: reliance on diffusion posterior sampling (DPS)-style value function approximations, which introduce non-negligible errors and lead to training instability and inference inefficiency. To overcome this, we introduce Physics-Informed Reward Fine-tuning (PIRF) — a method that bypasses value approximation by computing trajectory-level rewards and backpropagating their gradients directly. However, a naive implementation suffers from low sample efficiency and compromised data fidelity. PIRF mitigates these issues through two key strategies: (1) a layer-wise truncated backpropagation method that leverages the spatiotemporally localized nature of physics-based rewards, and (2) a weight-based regularization scheme that improves efficiency over traditional distillation-based methods. Across five PDE benchmarks, PIRF consistently achieves superior physical enforcement under efficient sampling regimes, highlighting the potential of reward fine-tuning for advancing scientific generative modeling. Our code is available at https://github.com/mingze-yuan/PIRF.

1 Introduction

Diffusion models [1, 2, 3] have emerged as powerful generative tools across modalities such as images [4], videos [5], and text [6]. Recently, their use has expanded into scientific machine learning [7, 8, 9], where the goal is to generate data governed by known physical laws—typically described by partial differential equations (PDEs). In this context, models must not only fit observed data but also satisfy physical constraints—a requirement we refer to as physical enforcement. However, standard diffusion models are trained purely with data-driven objectives and lack mechanisms to enforce such constraints, often resulting in physically invalid outputs [10, 11, 12].

Existing physics-informed diffusion models tackle this challenge by incorporating physical supervision during inference and/or training. Guidance-based methods[8, 13, 7] steer pretrained models using gradients from physics-informed losses[14], but typically require thousands of inference steps, making them computationally expensive. Training-based alternatives like physics-informed diffusion

*Corresponding authors.

Published in the AI for Science workshop (NeurIPS 2025).

model (PIDM) [15] inject physical loss during denoising steps, improving efficiency but often at the cost of stability and performance.

We introduce a new perspective on physics-informed generation by framing it as a sparse reward optimization problem, inspired by recent work that interprets diffusion sampling as a form of sequential decision-making [16, 17, 18]. In our formulation, rewards—derived from physical enforcement—are computed only at the final step of the diffusion trajectory. This viewpoint provides a unifying lens on prior methods: guidance-based approaches correspond to value-weighted sampling, where the gradient of an approximated value function is used to steer the inference process; meanwhile, PIDM [15] can be interpreted as enforcing a constant value function during training, avoiding the need to compute value gradients at inference time. However, both methods rely on diffusion posterior sampling (DPS)-style[19] value function approximations, which introduce significant errors in high-dimensional, non-convex PDE solution spaces[9]. These errors ultimately limit the stability and efficiency.

To overcome these limitations, we propose Physics-Informed Reward Fine-Tuning (PIRF)—a method that adapts reward fine-tuning (recently applied to text-to-image generation [18, 17]) to the physics-informed setting. PIRF avoids value approximation by directly computing trajectory-level rewards and backpropagating their gradients to fine-tune the model. However, a naive implementation leads to suboptimal data fidelity and inefficient sample usage. To address these issues, PIRF introduces two key innovations: First, we propose a layer-wise truncation strategy that updates only higher-resolution layers, motivated by the observation that physics-based rewards are often spatiotemporally localized. This improves training stability and data fidelity. Second, to improve sample efficiency, we replace costly distillation-based regularization with an offline weight-based regularizer, where we approximate the regularized effect using interpolated weights to greatly enhance efficiency.

We evaluate PIRF on five PDE benchmarks [7, 8], demonstrating consistent improvements over both guidance-based methods and PIDM in terms of physical enforcement. Notably, PIRF achieves strong performance under highly efficient inference regimes (e.g., 20 steps) and does not require gradient computations at test time. Ablation studies confirm the effectiveness of our design choices.

Our contributions are summarized as follows:

1. We offer a new perspective on physics-informed generation from reward optimization and identify a common bottleneck in prior approaches: value function approximation.
2. We introduce reward fine-tuning to the physics-informed setting and develop two strategies, layer-wise truncation and weight-based regularization.
3. We validate PIRF on five PDE benchmarks and demonstrate state-of-the-art performance in physical enforcement under efficient sampling regimes.

2 Background

Diffusion models [1, 2] generate data by progressively transforming a sample from a simple prior distribution—typically a standard Gaussian—into a sample drawn from the target data distribution $q(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathcal{X}$. This is accomplished via a fixed forward process that gradually adds Gaussian noise to a data sample $\boldsymbol{x}_0 \sim q(\boldsymbol{x})$ over T steps using a noise schedule $\{\beta_t \in (0,1)\}_{t=1}^T$:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}), \quad \text{with } q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1}, \beta_t \boldsymbol{I}). \tag{1}$$

To generate new samples, a learned reverse process approximates the true posterior $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ with a neural network parameterized by θ. Training is done by minimizing a simplified variational bound [1]. In this work, we use the denoising objective [3]:

$$\min_{\theta} \mathbb{E}_{\boldsymbol{x}_0 \sim q(\boldsymbol{x}), t, \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})} \left\| D_\theta(\boldsymbol{x}_t, t) - \boldsymbol{x}_0 \right\|_2^2, \tag{2}$$

where $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, and D_θ is the neural network used to predict the clean signal $\boldsymbol{x}_0$. For sampling, we start from $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$ and gradually sample $\boldsymbol{x}_{t-1} \sim q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. This conditional probability is not directly computable, and in practice, DDIM [20] samples $\boldsymbol{x}_{t-1}$ via

$$\boldsymbol{x}_{t-1} = \mu_\theta(\boldsymbol{x}_t, t) = \sqrt{\bar{\alpha}_{t-1}} D_\theta(\boldsymbol{x}_t, t) + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \frac{\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t} D_\theta(\boldsymbol{x}_t, t)}{\sqrt{1-\bar{\alpha}_t}} + \sigma_t \boldsymbol{\epsilon}, \tag{3}$$

where $\{\sigma_t\}_{t=1}^T$ are DDIM parameters, $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$. By choosing $\sigma_t = 0$, it reduces to a deterministic sampling process.

Physical laws are often formulated as partial differential equations (PDEs) over a domain $\Omega = \Omega_1 \times \Omega_2 \subset \mathbb{R}^d \times \mathbb{R}$, expressed as:

$$\boldsymbol{\mathcal{G}}\big[\boldsymbol{x}(\boldsymbol{\xi})\big] = 0, \quad \boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_d, \tau)^\top \in \Omega, \tag{4}$$

where $\boldsymbol{\mathcal{G}}$ denotes a differential operator encompassing the boundary and/or initial conditions, and $\boldsymbol{x}(\boldsymbol{\xi}) \in \mathbb{R}^c$ is the solution field that satisfies the set of PDEs over the domain Ω. Here, d denotes the spatial dimension and c the number of field components. The spatial domain is given by $\Omega_1 \subset \mathbb{R}^d$, and the time domain by $\Omega_2 \subset \mathbb{R}$. When the governing equation is time-independent, the problem reduces to a *static* PDE defined solely on the spatial domain $\Omega = \Omega_1$, with $\boldsymbol{\xi} = (\xi_1, \xi_2, \ldots, \xi_d)^\top$.

Note that, in general cases, the physical field can be decomposed into a solution field $\boldsymbol{u}(\boldsymbol{\xi}) \in \mathbb{R}^{c_1}$ and a PDE coefficient field $\boldsymbol{a}(\boldsymbol{\xi}) \in \mathbb{R}^{c_2}$ [8], which together characterize the physical system with $c = c_1 + c_2$. That is, we define $\boldsymbol{x} = [\boldsymbol{u}; \boldsymbol{a}]$ as the concatenation of the solution and coefficient fields. For instance, in the context of a 2D Darcy flow problem in porous media [7, 15], $\boldsymbol{x}$ describes the permeability and pressure fields, so $c = 2$ corresponds to these two fields, and $d = 2$ corresponds to the 2D spatial setting.

The *PDE residual* quantifies the discrepancy between candidate fields and the governing equations. It is defined as:

$$\boldsymbol{\mathcal{R}}(\boldsymbol{x}) = \boldsymbol{\mathcal{G}}[\boldsymbol{x}]. \tag{5}$$

For evaluation, we use the mean square error (MSE) between the PDE residual and zero, $\mathrm{MSE}(\boldsymbol{\mathcal{R}}(\boldsymbol{x}), \mathbf{0})$, as a scalar metric reflecting the degree of physical law enforcement.

3 Methodology

3.1 Problem statement

We aim to train a diffusion model whose samples comply with a set of governing PDEs, expressed as $\boldsymbol{\mathcal{R}}(\boldsymbol{x}) = \mathbf{0}$, where $\boldsymbol{\mathcal{R}}$ denotes the PDE residual operator. We formulate this problem from a reward optimization perspective, treating physical compliance as a reward function. The denoising process can be mapped to a Markov decision process (MDP) [21, 17] as follows (see Figure 4):

$$s_j \triangleq (t, \boldsymbol{x}_t), \quad a_j \triangleq \boldsymbol{x}_{t-1}, \quad \pi(a_j|s_j) \triangleq p_t(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t; \theta), \quad P(s_{j+1}|s_j, a_j) \triangleq (\delta_{t-1}, \delta_{\boldsymbol{x}_{t-1}}). \tag{6}$$

Here, $j = T - t$ is used for notational convenience, and δ_y denotes the Dirac delta distribution centered at y. The initial state distribution is defined as $\rho_0(s_0) = (\sigma_T, \mathcal{N}(0, \boldsymbol{I}))$. Each trajectory consists of T steps, after which the transition dynamics P lead to a terminal state. The reward is sparse and is only defined at the final state: $R(s_T, a_T) = r(\boldsymbol{x}_0)$. By defining the physics reward as the negative mean squared PDE residual, we obtain the following objective:

$$\theta^* = \underset{\{p_t(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t;\theta)\}_{t=0}^{T-1}}{\operatorname{argmax}} \mathbb{E}_{\{p_t\}}\left[r(\boldsymbol{x}_0)\right], \quad r(\boldsymbol{x}) \triangleq -\|\boldsymbol{\mathcal{R}}(\boldsymbol{x})\|_2^2. \tag{7}$$

3.2 Casting prior works as reward-based paradigms

This new perspective of reward optimization allows us to re-examine the prior physics-informed diffusion models. As summarized in Table 1, Prior works incorporate the physics constraints by reshaping the inference and/or training process with reward-based paradigms. A detailed discussion of related work is provided in subsection A.3.

Pure data-driven baselines select high-reward data from numerical simulations and train models directly on these samples. This strategy can be seen as a special case of *reward-weighted regression (RWR)*, where samples are filtered using a reward threshold and used in a weighted training objective:

$$\min_\theta \mathbb{E}_{\boldsymbol{x}_0, t, \boldsymbol{\epsilon}}\left[\omega(r(\boldsymbol{x}_0)) \cdot \|D_\theta(\boldsymbol{x}_t, t) - \boldsymbol{x}_0\|_2^2\right], \tag{8}$$

where the indicator function $\omega(r) = \mathbf{1}\left(r > \eta\right)$ selects samples with reward above threshold η. We denote the resulting model as $D_{\theta_{\text{base}}}$, used as the base model for guidance-based approaches.

Table 1: Casting physics-informd diffusion models into reward-based paradigms.

Approach	Reward-based paradigm	Inference efficiency	Value approx.	Physical Enforcement
Pure data-driven	Reward-weighted regression	High	No	Low
Guidance-based	Value-weighted sampling	Low	Yes	Medium
PG-Diffusion [7]	Reward gradient-conditioned CFG	Low	No	Medium
PIDM [15]	Constant value forcing	High	Yes	Medium
Ours	Reward backpropagation	High	No	High

Guidance-based methods [8, 13] adjust the inference process to steer it toward high-reward regions. Recent work [16] connects such methods to *value-weighted sampling* [4, 19], where the mean of the reverse transition is modified using the gradient of a value function:

$$\tilde{\boldsymbol{\mu}}_\theta(\boldsymbol{x}_t, t) = \boldsymbol{\mu}_\theta^{\text{base}}(\boldsymbol{x}_t, t) + \alpha \Sigma_t \nabla_{\boldsymbol{x}_t} v_t(\boldsymbol{x}_t), \tag{9}$$

where $v_t(\boldsymbol{x}_t)$ is the value function estimating expected reward from state $\boldsymbol{x}_t$:

$$v_t(\boldsymbol{x}_t) = \mathbb{E}_{\{p_\tau\}_{\tau=t}^1}\left[r(\boldsymbol{x}_0)|\boldsymbol{x}_t\right]. \tag{10}$$

In practice, v_t is approximated using a diffusion posterior sampling (DPS)-style method [19]:

$$v_t(\boldsymbol{x}_t) = \mathbb{E}[r(\boldsymbol{x}_0)|\boldsymbol{x}_t] \overset{(i)}{\approx} r(\mathbb{E}[\boldsymbol{x}_0|\boldsymbol{x}_t]) \overset{(ii)}{\approx} r(D_\theta(\boldsymbol{x}_t, t)). \tag{11}$$

Approximation (i) introduces a bias known as the Jensen gap [19, 22, 23], and approximation (ii) uses the denoising network as a point estimator, which can have high variance, particularly at early timesteps. These issues limit the physical precision of guidance-based methods. Moreover, the need to compute $\nabla_{\boldsymbol{x}_t} v_t(\boldsymbol{x}_t)$ slows down inference significantly.

Training-based methods aim to incorporate the physical constraints into the training process. For example, PG-Diffusion [7] uses the gradient of reward as condition for classifier-free guidance (CFG) [24]. **PIDM** [15] implicitly enforces a *constant value function* during training (termed as constant value forcing), avoiding computing gradients of v_t in Equation (9), thereby improving inference efficiency. Though PIDM is originally derived from virtual observables [25], we reinterpret it as encouraging

$$v_t(\boldsymbol{x}_t) \to \max_{\boldsymbol{x}\in\mathcal{X}} r(\boldsymbol{x}) = 0 \quad \forall \boldsymbol{x}_t, t, \tag{12}$$

based on the assumption that samples from numerical simulations attain the maximum reward (i.e., zero PDE residual). PIDM implements this value forcing via an augmented objective:

$$\min_\theta \mathbb{E}_{\boldsymbol{x}_0,t,\boldsymbol{\epsilon}}\left[\|D_\theta(\boldsymbol{x}_t;\sigma(t)) - \boldsymbol{x}_0\|_2^2 + \gamma_t \|v_t(\boldsymbol{x}_t) - 0\|_2^2\right]. \tag{13}$$

Since v_t is not directly available, PIDM approximates it via Equation (11), leading to:

$$\min_\theta \mathbb{E}_{\boldsymbol{x}_0,t,\boldsymbol{\epsilon}}\left[\|D_\theta(\boldsymbol{x}_t;\sigma(t)) - \boldsymbol{x}_0\|_2^2 + \gamma_t \|r(D_\theta(\boldsymbol{x}_t, t))\|_2^2\right], \tag{14}$$

which matches the training objective in PIDM. Although a two-step DDIM estimation is proposed to reduce the error in approximation (ii) (from Equation (11)), the bias from approximation (i) remains unaddressed. Consequently, due to approximation and optimization errors, value forcing cannot be perfectly achieved, which limits the effectiveness of the method.

Limitations. Pure data-driven baselines lack explicit physical constraints and generally perform poorly in terms of physical enforcement. Both guidance-based approaches and PIDM rely heavily on *value function approximation*, typically implemented via Equation (11). While this strategy is effective in some inverse problems [19], we empirically observe that it introduces substantial error in physics-informed generation—especially under efficient sampling regimes (see Figure 5). These limitations motivate us to sidestep value function approximation.

3.3 PIRF: Physics-Informed Reward Fine-Tuning

Motivated by recent work on text-to-image diffusion alignment [18, 26], we introduce reward fine-tuning into the physics-informed setting, termed as Physics-Informed Reward Fine-Tuning (PIRF).

Algorithm 1 A high-level framework of reward backpropagation

Require: Pre-trained model $\{p_t(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t;\theta^{\text{base}})\}_{t=0}^{N-1}$, batch size m, learning rate γ, physics-informed reward function $r(\cdot)$ (defined in Equation (7)), number of iterations K

1: Initialize parameters: $\theta_1 \leftarrow \theta^{\text{base}}$
2: **for** $k = 1$ **to** K **do**
3: Sample m trajectories $\{\boldsymbol{x}_t^{(i)}(\theta_k)\}_{t=0}^{N}$ from current model $\{p_t(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t;\theta_k)\}_{t=T}^{1}$
4: Update parameters: $\theta_{k+1} \leftarrow \theta_k + \gamma\nabla_\theta \left[\frac{1}{m}\sum_{i=1}^{m} r(\boldsymbol{x}_0^{(i)}(\theta_k))\right]\Big|_{\theta=\theta_s}$
5: **end for**
6: **return** fine-tuned $\{p_t(\cdot \mid \cdot;\theta_K)\}_{t=0}^{N-1}$

Overview. As shown in Figure 1 and Algorithm 1, PIRF starts with a pre-trained model θ_{base} via standard diffusion training [3]. During fine-tuning, it bypasses value function approximation by iterating over two phases: (1) sampling full diffusion trajectories (line 3), (2) fine-tuning the model using accurate reward signals (line 4). Since physics-based rewards are differentiable, we directly backpropagate gradients of the final-step reward $r(\boldsymbol{x}_0)$ through the entire denoising trajectory [18].

For notational simplicity, we formulate it with the deterministic sampling setting, where trajectories begin from $\boldsymbol{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$ and evolve as:

$$\boldsymbol{x}_{t-1} = \mathcal{F}_\theta^{(t)}(\boldsymbol{x}_t) \triangleq \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \quad t = 1, \cdots, T, \tag{15}$$

leading to a final state:

$$\boldsymbol{x}_0 = \mathcal{F}_\theta^{(1)} \circ \cdots \circ \mathcal{F}_\theta^{(T)}(\boldsymbol{x}_T). \tag{16}$$

While conceptually straightforward, this approach is sample inefficient as it requires computing full trajectories for every update. Additionally, it introduces two major challenges: (1) memory overhead from storing all intermediate states during backpropagation; and (2) *reward hacking*, where the model sacrifices data fidelity or diversity to over-optimize the reward. These challenges motivate the need for practical improvements.

Step-wise truncation baseline. Prior work such as DRaFT [18] addresses memory inefficiency via step-wise truncation, limiting backpropagation to the last K steps. PalSB [9] extends this to a physics-aligned Schrödinger bridge model. We adopt this approach as a baseline (termed PIRF-base), but observe that in physics-informed generation, step-wise truncation still allows excessive model flexibility, leading to overfitting and reward hacking.

Layer-wise truncation (LT). To further constrain optimization, we introduce a finer-grained truncation strategy at the network layer level. Specifically, the denoising operator $\mathcal{F}_\theta^{(t)}(\boldsymbol{x}_t)$ is composed of M neural network layers:

$$\mathcal{F}_\theta^{(t)}(\boldsymbol{x}_t) = \mathcal{F}_{\theta,M}^{(t)} \circ \mathcal{F}_{\theta,M-1}^{(t)} \circ \cdots \circ \mathcal{F}_{\theta,1}^{(t)} \circ \boldsymbol{x}_t. \tag{17}$$

Unlike global rewards used in text-to-image models [21, 17] (e.g., CLIP score), physics-based rewards—such as PDE residuals—are inherently local, typically involving only neighboring points via finite difference schemes (see Algorithm A.1 for an example).

Motivated by this locality, we restrict parameter updates to only the final m layers corresponding to the highest-resolution stage, i.e., $\{\mathcal{F}_{\theta,M}, \mathcal{F}_{\theta,M-1}, \ldots, \mathcal{F}_{\theta,M-m+1}\}$. In U-Net architectures, lower-resolution layers capture global, low-frequency semantics, while higher-resolution layers encode local, high-frequency details. By freezing the low-resolution layers, we constrain the model's global semantics while allowing sufficient flexibility to satisfy localized physical constraints. Empirically, this strategy stabilizes training, mitigates reward hacking, and improves sample efficiency.

Weight regularization (WR). Conventional text-to-image reward fine-tuning methods [21, 17, 16] use distillation-based regularization to prevent reward hacking, typically via a KL divergence between the current and base model policies:

$$\max_\theta \mathbb{E}\left[r(\boldsymbol{x}_0) - \lambda_{\text{distill}} \sum_{t=0}^{N-1} \|D_\theta(\boldsymbol{x}_t, t) - D_\theta^{\text{base}}(\boldsymbol{x}_t, t)\|_2^2\right], \tag{18}$$

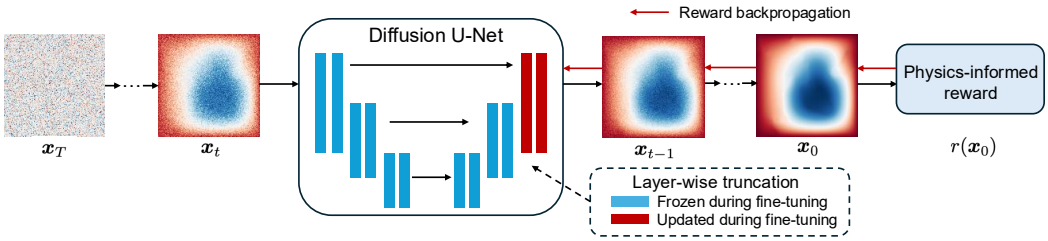

Figure 1: PIRF bypasses the challenges of value function approximation by directly backpropagating gradients from accurate final-state rewards. The proposed layer-wise truncation strategy, which updates only high-resolution layers, further enhances training stability.

However, this regularization doubles the number of forward passes, significantly reducing sample efficiency. Therefore, we re-examine the need for distillation under physics-based rewards. Noting that distillation objectives can be approximated under mild assumptions, we propose a weight-space alternative. If D_θ is Lipschitz-continuous with respect to θ (this is reasonable due to the normarlized inputs in practice), the KL term can be relaxed to a simple weight penalty:

$$\max_{\theta} \mathbb{E}\left[r(\boldsymbol{x}_0) - \lambda_{\text{weight}} \|\theta - \theta_{\text{base}}\|_2^2\right]. \tag{19}$$

We call this *online weight regularization* (ON-WR). To further improve efficiency, we explore two lightweight, *offline* strategies (OFF-WR). First, we observe that *early stopping* can effectively limit weight drift, thereby mitigating reward hacking. Second, we apply *linear interpolation* between the base and fine-tuned model, implemented as an exponential moving average (EMA) during fine-tuning. This implicit regularization approximates Equation (19) post-hoc and offers additional sample efficiency.

4 Experiments

4.1 Experimental setup

Datasets. We validate our approach on five PDE datasets following [8, 7], including Burgers' equation, Darcy flow, the inhomogeneous Helmholtz equation, the Poisson equation, and Kolmogorov flow. To ensure a fair comparison, we use the publicly available datasets provided in [8, 7]. We focus on the unconditional setting as [15], where the goal is to generate the entire physical fields, including both the PDE solution field and coefficient field. For example, in Darcy flow, we generate the concatenation of permeability (coefficient) and pressure (solution) fields. This setting can be naturally extended to conditional generation by training a conditional model. Detailed descriptions of each PDE and the data preparation process are provided in Appendix subsection A.2, and a summary of the datasets is presented in Table 5.

Implementation details. We implement PIRF on top of the EDM [3] framework. For consistency, we use base models from DiffusionPDE for most datasets. For Kolmogorov flow, which differs in its PDE setup, we train a base model from scratch using the official EDM configurations. We use a learning rate of 0.001 for pretraining and 0.0001 for fine-tuning. Fine-tuning is initially performed using 80 sampling steps, and we observe that the model generalizes well to 40-step inference. However, performance drops at 20 steps, so we fine-tune a separate model under the 20-step setting. We follow DRaFT [18] to truncate the last step for 80-step sampling and the last two steps for 20-step sampling to save memory. Fine-tuning on approximately 180k trajectories is sufficient for strong reward optimization, using around 15h for 80-step fine-tuning and 5h for 20-step fine-tuning with two NVIDIA 80GB A100 GPUs. The EMA half-life is set as 50000.

Baselines. We compare PIRF with several state-of-the-art physics-informed diffusion models, including DiffusionPDE [8], CoCoGen [13], PG-Diffusion [7], and PIDM [15]. All models are implemented using the EDM [3] framework for consistency. DiffusionPDE applies DPS [19] to guide the reverse process, while CoCoGen further adds post-sampling correction steps. PG-Diffusion adopts classifier-free guidance (CFG) [24], where the gradient of the PDE loss serves as the conditioning

Table 2: Comparison of PDE residual mean squared error (MSE ↓) across different sampling steps.

# Steps	Method	Burgers	Darcy	Helmholtz	Poisson	Kolmogorov
	Training data	1.06E-02	16.44	1.10	1.20	14.00
80	EDM [3]	2.01E-01	18.34	2.88	4.84	206.16
	DiffusionPDE [8]	8.38E-03	7.99	2.14	1.21	51.04
	CoCoGen [13]	4.62E-03	4.56	0.78	0.95	76.66
	PG-Diffusion [7]	1.45E-01	7.97	7.81	8.65	408.11
	PIDM [15]	1.67E-01	14.23	1.33	2.87	135.20
	Ours	**1.68E-03**	**1.29**	**0.17**	**0.19**	**19.28**
40	EDM [3]	2.08E-01	25.43	9.71	5.72	205.26
	DiffusionPDE [8]	2.26E-02	22.73	8.71	3.02	84.53
	CoCoGen [13]	8.69E-03	8.25	1.63	1.58	80.32
	PG-Diffusion [7]	1.45E-01	8.00	7.87	8.48	368.39
	PIDM [15]	1.73E-01	13.13	1.31	2.85	136.79
	Ours	**2.31E-03**	**1.70**	**0.18**	**0.20**	**19.37**
20	EDM [3]	2.18E-01	28.94	5.20	5.07	219.40
	DiffusionPDE [8]	5.60E-02	15.65	2.16	2.11	125.52
	CoCoGen [13]	**1.51E-02**	7.16	0.75	1.04	101.27
	PG-Diffusion [7]	1.32E-01	10.42	7.51	7.90	299.34
	PIDM [15]	1.83E-01	9.45	1.19	2.56	143.84
	Ours	3.75E-02	**1.99**	**0.28**	**0.28**	**20.62**

Table 3: Inference efficiency of different methods using Euler solver with N steps and optional M post-corrections. NRQ: number of reward queries. NBM: number of backpropagating model to compute reward gradient.

Method	NRQ ↓	NBM ↓
PG-Diffusion [7]	N	N
DiffusionPDE [8]	N	N
CoCoGen [13]	$N + M$	N
PIDM [15]	0	0
Ours	0	0

Table 4: Sample efficiency analysis. T: total sample steps, K: steps truncated. Time: used seconds for fine-tuning 1k trajectories. LT: layer-wise truncation; WR: weight regularization. ON: online, OFF: offline.

T	K	LT	WR	Time (s)
80	2		OFF	331
80	1		OFF	180
20	2		ON	163
20	2		OFF	87
20	2	✓	OFF	84

signal. Closer to our approach, PIDM incorporates a physics-informed supervision loss on the denoised output during training. For DiffusionPDE and CoCoGen, we follow the standard setup from DiffusionPDE by applying physics guidance during only the last 20% of sampling steps. We perform grid searches over guidance scale and the number of correction steps and report the best results. For PG-Diffusion and PIDM, since their original papers do not cover all benchmarks used here, we extend their implementation to the new datasets using their official configurations. For PIDM, we use the mean estimation mode and initialize from the base model when training from scratch fails to converge due to instability.

Evaluation metrics. All methods are evaluated using the Heun sampler with the default schedule from EDM using 20, 40, and 80 steps. We generate 1600 samples with same random seeds and compute the average PDE residual MSE as the quantitative metric. Furthermore, we visually checked whether samples are physcially plausible and diversified, in case of reward hacking happens in PIRF. We also compare their inference efficiency defined by number of reward function querying (NRQ), and number of backpropagating the model for gradient computation (NBM).

4.2 Results

Main results. Table 2 compares PIRF with baseline methods across five PDE benchmarks using 80, 40, and 20 sampling steps. PIRF consistently outperforms all baselines in terms of physical precision, except on the Burgers' equation with 20 steps, where it slightly trails behind CoCoGen [13]. This may be attributed to the simplicity of the 1D Burgers' equation, where CoCoGen's post-correction

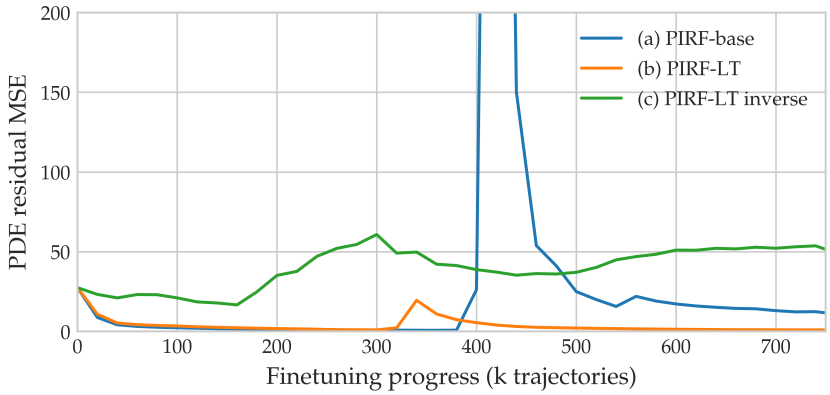

Figure 2: Effect of layer-wise truncation (LT). PIRF-LT (only updating high-resolution layer) achieves more stable long-term training compared to PIRF-base. As a inverse design, variant (c) fails to converge by only updating low-resolution layers, validating our assertion on the localized property of physics rewards. Additionally, PIRF-base exhibits signs of reward hacking while PIRF stays stable, as further illustrated in Figure 6

.

procedure is particularly effective. However, on more complex PDEs, PIRF achieves significantly better performance than all baselines. Furthermore, PIRF not only surpasses baseline models but also matches or exceeds the physical precision of the training data itself, which serves as a "gold standard" derived from conventional simulators. In terms of inference efficiency, we evaluate methods based on the number of reward function queries (NRQ) and the number of model backward passes (NBM), as shown in Table 3. PIRF and PIDM [15] are the most efficient at inference time, but PIRF achieves superior physical precision. Although CoCoGen is the second-best method in terms of physical enforcement across most benchmarks, it incurs the highest inference cost among all evaluated methods.

Effect of layer-wise truncation. We investigate the impact of layer-wise truncation (LT) in PIRF on the Darcy flow dataset. In our method, only the high-resolution decoder layers are updated during fine-tuning (PIRF-LT). We compare this against two baselines: (i) updating all layers (PIRF-base) and (ii) updating only the low-resolution decoder layers (PIRF-LT inverse). As shown in Figure 2, PIRF-LT maintains stable performance over long-term training and is robust to minor oscillations. In contrast, PIRF-base exhibits significant fluctuations and, once degraded, fails to recover. To better understand this behavior, we visualize the output fields at the 600k training step in Figure 6. We observe that variant (i), which updates all layers, produces unnatural void regions—an indication of reward hacking and deviation from the true data distribution. In contrast, PIRF with LT preserves both training stability and distributional fidelity. For further comparison, we evaluate variant (ii), where only low-resolution layers are updated. This variant fails to converge, supporting our hypothesis that physics-based rewards are predominantly local and are most effectively optimized through high-resolution features. These results suggest that updating only high-resolution decoder layers is not only sufficient but also preferable for achieving high physical fidelity and stable training.

Effect of weight regularization. The impact of weight regularization (WR) in PIRF is illustrated in Figure 3. PIRF without WR (column b) achieves higher physical reward but significantly deviates from the base model's distribution (column a), exhibiting clear signs of reward hacking. In contrast, PIRF with WR (column c) achieves a more favorable balance between physical enforcement and data fidelity. Moreover, offline WR enhances sample efficiency compared to online WR. As shown in Table 4, compared to online WR, offline WR reduces the fine-tuning time for 1,000 trajectories from approximately 163 seconds to 87 seconds—a nearly 2× improvement.

5 Conclusion

In this paper, we proposed PIRF, a physics-informed reward fine-tuning framework for diffusion models. Validated on five PDE benchmarks, PIRF achieves high physical precision under efficient

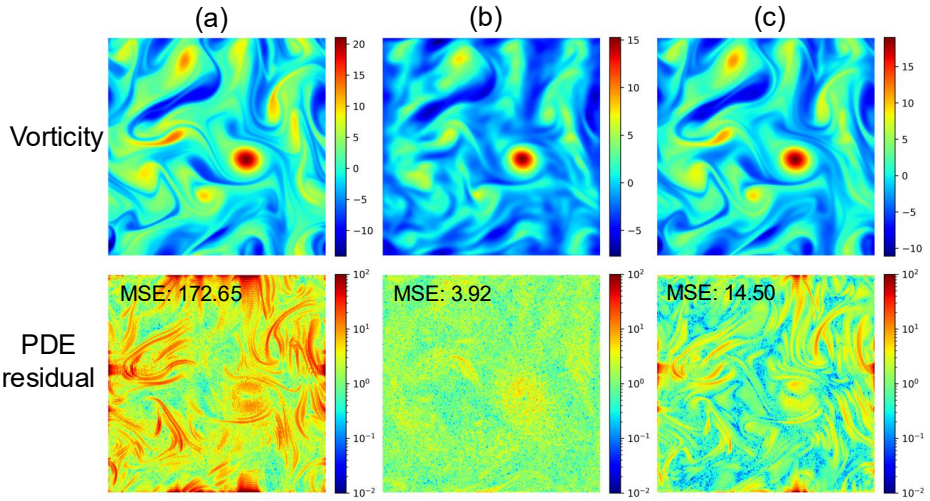

Figure 3: Effectiveness of weight regularization in mitigating reward hacking. From left to right: (a) sample from the base model on Kolmogorov flow, (b) PIRF without weight regularization, and (c) PIRF with offline weight regularization. While (b) achieves lowest PDE residual MSE, it introduces distortions in the vorticity field, indicating reward hacking. In contrast, (c) preserves structural consistency with the base model while maintaining low PDE residual.

sampling regimes. The introduction of layer-wise truncation and weight regularization further enhances both performance and training stability. We hope this work highlights the potential of reward fine-tuning as a principled approach to incorporating physical priors into generative models, bridging the gap between data-driven and model-based approaches in scientific domains.

Acknowledgements

This work was supported by the National Science Foundation AI Institute (NSF Award No. 2112085) and by the National Institutes of Health (NIH Award No. R01HL159183).

References

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

[2] 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.

[3] 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.

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

[5] 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.

[6] Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025.

[7] 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.

[8] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. DiffusionPDE: Generative PDE-solving under partial observation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[9] Zeyu Li, Hongkun Dou, Shen Fang, Wang Han, Yue Deng, and Lijun Yang. Physics-aligned field reconstruction with diffusion bridge. In *The Thirteenth International Conference on Learning Representations*, 2025.

[10] Haixin Wang, Yadi Cao, Zijie Huang, Yuxuan Liu, Peiyan Hu, Xiao Luo, Zezheng Song, Wanjia Zhao, Jilin Liu, Jinan Sun, et al. Recent advances on machine learning for computational fluid dynamics: A survey. *arXiv preprint arXiv:2408.12171*, 2024.

[11] Lizao Li, Robert Carver, Ignacio Lopez-Gomez, Fei Sha, and John Anderson. Generative emulation of weather forecast ensembles with diffusion models. *Science Advances*, 10(13):eadk4489, 2024.

[12] Wei Qian, Gaoji Su, Dan Guo, Jinxing Zhou, Xiaobai Li, Bin Hu, Shengeng Tang, and Meng Wang. Physdiff: Physiology-based dynamicity disentangled diffusion model for remote physiological measurement. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, pages 6568–6576, 2025.

[13] Christian Jacobsen, Yilin Zhuang, and Karthik Duraisamy. Cocogen: Physically consistent and conditioned score-based generative models for forward and inverse problems. *SIAM Journal on Scientific Computing*, 47(2):C399–C425, 2025.

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

[15] Jan-Hendrik Bastek, WaiChing Sun, and Dennis Kochmann. Physics-informed diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[16] Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024.

[17] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

[18] Kevin Clark, Paul Vicol, Kevin Swersky, and David J. Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *The Twelfth International Conference on Learning Representations*, 2024.

[19] 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.

[20] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[21] Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. Dpok: Reinforcement learning for fine-tuning text-to-image diffusion models. *Advances in Neural Information Processing Systems*, 36:79858–79885, 2023.

[22] Xiang Gao, Meera Sitharam, and Adrian E Roitberg. Bounds on the jensen gap, and implications for mean-concentrated distributions. *arXiv preprint arXiv:1712.05267*, 2017.

[23] Slavko Simic. On a global upper bound for jensen's inequality. *Journal of mathematical analysis and applications*, 343(1):414–419, 2008.

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

[25] Maximilian Rixner and Phaedon-Stelios Koutsourelakis. A probabilistic generative model for semi-supervised training of coarse-grained surrogates and enforcing physical constraints through virtual observables. *Journal of Computational Physics*, 434:110218, 2021.

[26] Xiaoshi Wu, Yiming Hao, Manyuan Zhang, Keqiang Sun, Zhaoyang Huang, Guanglu Song, Yu Liu, and Hongsheng Li. Deep reward supervisions for tuning text-to-image diffusion models. In *European Conference on Computer Vision*, pages 108–124. Springer, 2024.

[27] Gary J Chandler and Rich R Kerswell. Invariant recurrent solutions embedded in a turbulent two-dimensional kolmogorov flow. *Journal of Fluid Mechanics*, 722:554–595, 2013.

[28] Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 1(3):1–27, 2024.

[29] Kimin Lee, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu. Aligning text-to-image models using human feedback. *arXiv preprint arXiv:2302.12192*, 2023.

[30] Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*, 2023.

[31] Fei Deng, Qifei Wang, Wei Wei, Tingbo Hou, and Matthias Grundmann. Prdp: Proximal reward difference prediction for large-scale reward finetuning of diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7423–7433, June 2024.

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

[33] Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Sergey Levine, and Tommaso Biancalani. Feedback efficient online fine-tuning of diffusion models. *arXiv preprint arXiv:2402.16359*, 2024.

[34] Bradley Efron. Tweedie's formula and selection bias. *Journal of the American Statistical Association*, 106(496):1602–1614, 2011.

[35] Georg Kohl, Li-Wei Chen, and Nils Thuerey. Benchmarking autoregressive conditional diffusion models for turbulent flow simulation. *arXiv preprint arXiv:2309.01745*, 2023.

[36] Gefan Yang and Stefan Sommer. A denoising diffusion model for fluid field prediction. *arXiv preprint arXiv:2301.11661*, 2023.

[37] Salva Rühling Cachay, Bo Zhao, Hailey Joren, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *Advances in neural information processing systems*, 36:45259–45287, 2023.

[38] Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. Pde-refiner: Achieving accurate long rollouts with neural pde solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

[39] Chaoran Cheng, Boran Han, Danielle C. Maddix, Abdul Fatir Ansari, Andrew Stuart, Michael W. Mahoney, and Bernie Wang. Gradient-free generation for hard-constrained systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

[40] Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, pages 3934–3942. PMLR, 2024.

[41] François Rozet and Gilles Louppe. Score-based data assimilation. *Advances in Neural Information Processing Systems*, 36:40521–40541, 2023.

[42] Yongquan Qu, Juan Nathaniel, Shuolin Li, and Pierre Gentine. Deep generative data assimilation in multimodal setting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 449–459, 2024.

[43] David Ruhe, Jonathan Heek, Tim Salimans, and Emiel Hoogeboom. Rolling diffusion models. In *Forty-first International Conference on Machine Learning*, 2024.

[44] Long Wei, Peiyan Hu, Ruiqi Feng, Haodong Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Diffphycon: A generative approach to control complex physical systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[45] Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard Turner, and Emile Mathieu. On conditional diffusion models for pde simulations. *Advances in Neural Information Processing Systems*, 37:23246–23300, 2024.

[46] Yulai Zhao, Masatoshi Uehara, Gabriele Scalia, Sunyuan Kung, Tommaso Biancalani, Sergey Levine, and Ehsan Hajiramezanali. Adding conditional control to diffusion models with reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[47] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023.

A Technical Appendices and Supplementary Material

A.1 More on methodology

Analysis on value function approximation errors Figure 5 shows the distribution of value approximation errors across sampling steps. Given a trajectory $\{\boldsymbol{x}_t\}_{t=T}^{0}$, the y-axis (log scale) represents the absolute error of value approximation, defined as $|r(\hat{\boldsymbol{x}}_0^t) - r(\boldsymbol{x}_0)|$, where $\hat{\boldsymbol{x}}_0^t = D_\theta(\boldsymbol{x}_0, t)$ is the point estimate of final state given $\boldsymbol{x}_t$. The x-axis corresponds to the step index $t+1$, with $T = 20$ chosen to reflect efficient sampling settings. We use the base model from DiffusionPDE [8] for Darcy flow to plot this graph with 800 random samples. We can find that value function approxmation errors remains high even in late steps, leading to a key bottleneck for existing approaches. Motivated by this, we use physics-informed reward backpropagation to bypass this bottleneck.

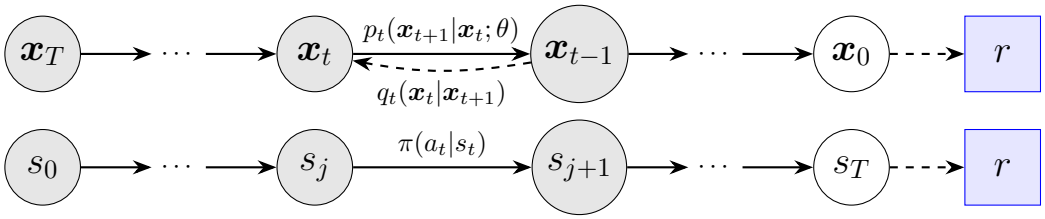

Figure 4: Formulating diffusion model fine-tuning as a sequential decision-making problem with sparse reward. Here $j + t = T$, The top row illustrates the reverse generative process; the bottom row shows its equivalent MDP.

Algorithm 2 Pseudocode of computing Burgers' equation residual in a PyTorch-like style.

```
class BurgersResidual(nn.Module):
    def __init__(self, domain_length=1.0, pixels_per_dim=128):
        super().__init__()
        dx = domain_length / pixels_per_dim
        dt = domain_length / pixels_per_dim
        self.deriv_x = tensor([[-1, 0, 1]]).view(1,1,1,3) / (2 * dx)
        self.deriv_t = tensor([[-1], [0], [1]]).view(1,1,3,1) / (2 *
            dt)

    def forward(self, u):
        # u: input of shape (B, 1, T, X)
        # u partial on x
        u_x = pad(u, (1,1,0,0), mode='replicate')
        u_x = conv2d(u_x, self.deriv_x)

        # u partial on t
        u_t = pad(u, (0,0,1,1), mode='replicate')
        u_t = conv2d(u_t, self.deriv_t)

        # u_x partial on x
        u_xx = pad(u_x, (1,1,0,0), mode='replicate')
        u_xx = conv2d(u_xx, self.deriv_x)

        # compute residual
        return u_t + u * u_x - 0.01 * u_xx
```

A.2 Details on benchmarks

A summary of benchmarks is provided in Table 5. Here we illustrate dataset simulation details for each PDE. For each PDE, if not specified, we generate $50,000$ samples.

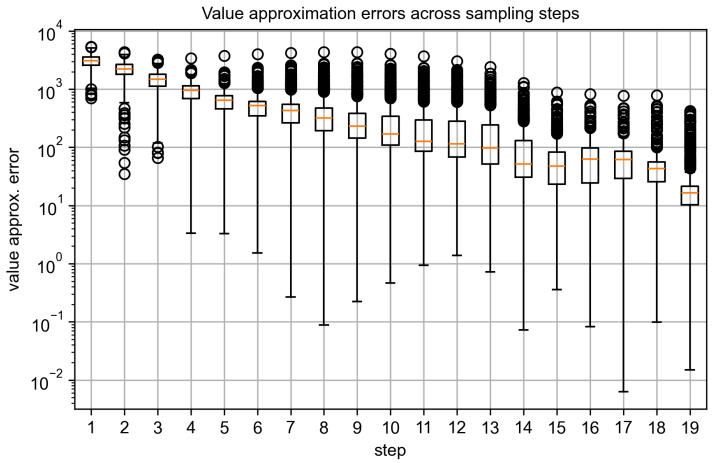

Figure 5: Box plot showing the distribution of value approximation errors across sampling steps in existing approaches. Notably, the approximation error remains high even in late sampling steps, highlighting a key bottleneck in current physics-informed diffusion models.

A.2.1 Darcy flow

Darcy flow equations [13] describe the relationship between fluid pressure and the permeability of a porous medium. A coefficient field $\boldsymbol{a}(\boldsymbol{\xi})$ characterizes the ease of fluid flow at each spatial location $\boldsymbol{\xi} \in \Omega$, while a source function $f(\boldsymbol{\xi})$ represents fluid injection or extraction. The pressure field $\boldsymbol{u}(\boldsymbol{\xi})$ and velocity field $\boldsymbol{v}(\boldsymbol{\xi})$ satisfy:

$$\begin{aligned} \boldsymbol{v}(\boldsymbol{\xi}) &= -\boldsymbol{a}(\boldsymbol{\xi})\nabla \boldsymbol{u}(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \Omega, \\ \nabla \cdot \boldsymbol{v}(\boldsymbol{\xi}) &= f(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \Omega, \\ \boldsymbol{v}(\boldsymbol{\xi}) \cdot \hat{\boldsymbol{n}}(\boldsymbol{\xi}) &= 0, \quad \boldsymbol{\xi} \in \partial\Omega, \\ \int_{\mathcal{X}} \boldsymbol{u}(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} &= 0. \end{aligned} \tag{20}$$

$$f(\boldsymbol{\xi}) = \begin{cases} r, & \left|\xi_i - \frac{1}{2}w\right| \le \frac{1}{2}w, \quad i = 1, 2, \\ -r, & \left|\xi_i - 1 + \frac{1}{2}w\right| \le \frac{1}{2}w, \quad i = 1, 2, \\ 0, & \text{otherwise.} \end{cases} \tag{21}$$

We follow the settings from [8], a constant source is used: $f(\boldsymbol{\xi}) = 1$. We first generate Gaussian random fields on $(0, 1)^2$ with $\mu \sim \mathcal{N}(0, (\Delta + 9\boldsymbol{I})^{-2})$, and then define:

$$\boldsymbol{a}(\xi) = \begin{cases} 12, & \text{if } \mu(\xi) \ge 0, \\ 3, & \text{if } \mu(\xi) < 0. \end{cases} \tag{22}$$

The equations are solved using finite difference approximations on a spatial domain $\mathcal{X} = [0, 1]^2$ with an $n \times n$ grid [13] (where $n = 128$). Each grid point indexed by i, j corresponds to:

$$\boldsymbol{\xi}_{i,j} = \left[\frac{i-1}{n-1}, \frac{j-1}{n-1}\right]^\top, \quad i, j = 1, \ldots, n. \tag{23}$$

Table 5: Summary of PDE datasets.

Dataset	Spatial resolution	Temporal resolution	# Samples	Source
Burgers' equation	128	128	50,000	[8]
Darcy flow	128×128	N/A	50,000	[8]
Helmholtz equation	128×128	N/A	50,000	[8]
Poisson equation	128×128	N/A	50,000	[8]
Kolmogorov flow	256×256	320	40	[7]

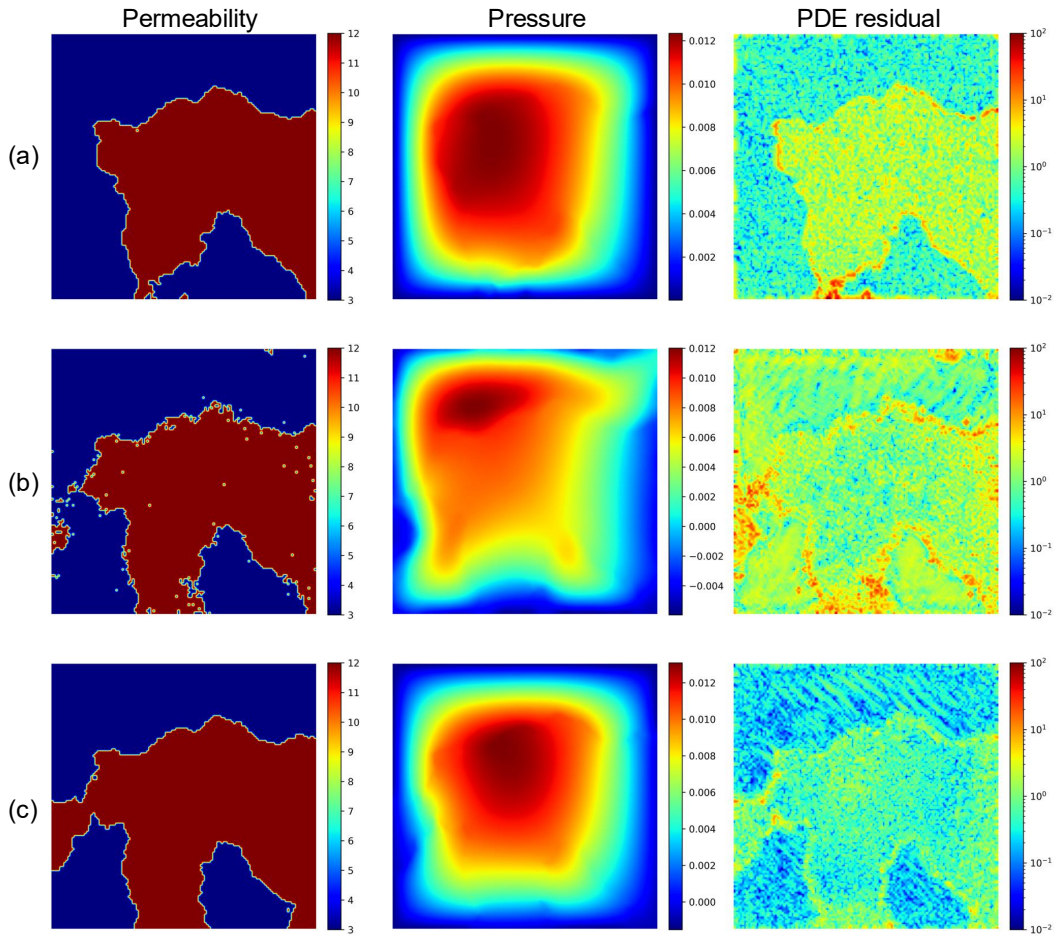

Figure 6: Illustrative example at the 600k fine-tuning checkpoint on Darcy flow. From top to bottom: (a) base model, (b) PIRF without layer-wise truncation, (c) PIRF with layer-wise truncation. In (b), the permeability field exhibits artificial voids within originally intact regions, reflecting a divergence from the data distribution driven by over-optimization of the physics reward—an instance of reward hacking. In contrast, (c) shows that layer-wise truncation preserves structural integrity, yielding more stable and physically consistent outputs.

The solution $\boldsymbol{u}(\boldsymbol{\xi})$ is vectorized as $\boldsymbol{u} \in \mathbb{R}^{n^2}$, yielding a linear system $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$, where $\boldsymbol{A} \in \mathbb{R}^{(n^2+1)\times n^2}$. The final row enforces the integral constraint $\int_{\mathcal{X}} \boldsymbol{u}(\boldsymbol{\xi})\,\mathrm{d}\boldsymbol{\xi} = 0$. Derivatives are approximated using finite differences, and boundary conditions are applied by adjusting stencils. The velocity is recovered as $\boldsymbol{v}(\boldsymbol{\xi}) = -\boldsymbol{a}(\boldsymbol{\xi})\nabla\boldsymbol{u}(\boldsymbol{\xi})$, with gradients estimated using second-order central differences.

To assess physical consistency, we compute the residual:

$$\boldsymbol{\mathcal{R}}(\boldsymbol{u},\boldsymbol{a}) = \boldsymbol{a}(\boldsymbol{\xi})\frac{\partial^2 \boldsymbol{u}(\boldsymbol{\xi})}{\partial \xi_1^2} + \frac{\partial \boldsymbol{a}(\boldsymbol{\xi})}{\partial \xi_1}\frac{\partial \boldsymbol{u}(\boldsymbol{\xi})}{\partial \xi_1} + \boldsymbol{a}(\boldsymbol{\xi})\frac{\partial^2 \boldsymbol{u}(\boldsymbol{\xi})}{\partial \xi_2^2} + \frac{\partial \boldsymbol{a}(\boldsymbol{\xi})}{\partial \xi_2}\frac{\partial \boldsymbol{u}(\boldsymbol{\xi})}{\partial \xi_2} + f(\boldsymbol{\xi}). \tag{24}$$

The residual is evaluated at each grid point. The integral constraint is enforced by normalizing: $\boldsymbol{u} = \tilde{\boldsymbol{u}} - \int_{\mathcal{X}} \tilde{\boldsymbol{u}}\,\mathrm{d}\boldsymbol{\xi}$, ensuring the residual reflects only PDE compliance.

A.2.2 Inhomogeneous Helmholtz equation

We consider the static inhomogeneous Helmholtz equation with no-slip boundary conditions on $\partial\Omega$, describing wave propagation:

$$\begin{aligned} \nabla^2 \boldsymbol{u}(\boldsymbol{\xi}) + k^2 \boldsymbol{u}(\boldsymbol{\xi}) = \boldsymbol{a}(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in \Omega, \\ \boldsymbol{u}(\boldsymbol{\xi}) = 0, \quad \boldsymbol{\xi} \in \partial\Omega. \end{aligned} \tag{25}$$

Here, k is a constant. When $k = 0$, Equation 25 reduces to the Poisson equation. We set $k = 1$ for the Helmholtz case. The residual is computed as:

$$\boldsymbol{\mathcal{R}}(\boldsymbol{u}, \boldsymbol{a}) = \nabla^2 \boldsymbol{u}(\boldsymbol{\xi}) + k^2 \boldsymbol{u}(\boldsymbol{\xi}) - \boldsymbol{a}(\boldsymbol{\xi}). \tag{26}$$

Following [8], we first generate Gaussian fields on $(0, 1)^2$ with $\boldsymbol{a} \sim \mathcal{N}(0, (\Delta + 9\boldsymbol{I})^{-2})$, then solve for $\boldsymbol{u}$ using second-order finite differences. To enforce the no-slip boundary condition, we multiply the solution by a mollifier $\sin(\pi\xi_1)\sin(\pi\xi_2)$, where $\boldsymbol{\xi} = (\xi_1, \xi_2) \in (0, 1)^2$. Both $\boldsymbol{a}$ and $\boldsymbol{u}$ are defined on 128×128 grids.

A.2.3 Kolmogorov flow

The dataset used in this study is based on the two-dimensional Kolmogorov flow [27], governed by the incompressible Navier–Stokes equations in vorticity form:

$$\begin{aligned} \frac{\partial \boldsymbol{u}(\boldsymbol{\xi}, \tau)}{\partial \tau} + \boldsymbol{\nu}(\boldsymbol{\xi}, \tau) \cdot \nabla \boldsymbol{u}(\boldsymbol{\xi}, \tau) = \frac{1}{Re} \nabla^2 \boldsymbol{u}(\boldsymbol{\xi}, \tau) + f(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in (0, 2\pi)^2,\ \tau \in (0, T], \\ \nabla \cdot \boldsymbol{\nu}(\boldsymbol{\xi}, \tau) = 0, \quad \boldsymbol{\xi} \in (0, 2\pi)^2,\ \tau \in (0, T], \\ \boldsymbol{u}(\boldsymbol{\xi}, 0) = \boldsymbol{u}_0(\boldsymbol{\xi}), \quad \boldsymbol{\xi} \in (0, 2\pi)^2. \end{aligned} \tag{27}$$

Here, $\boldsymbol{u}$ denotes the vorticity field, $\boldsymbol{\nu}$ is the velocity field, and Re is the Reynolds number, set to 1000. The function $f(\boldsymbol{\xi})$ represents the external forcing term, and $\boldsymbol{\xi} = [\xi_1, \xi_2]$ is the spatial coordinate. We follow the settings in [7], where periodic boundary conditions are applied. The forcing used for the 2D Kolmogorov flow is defined as: $f(\boldsymbol{\xi}) = -4\cos(4\xi_2) - 0.1\,\boldsymbol{u}(\boldsymbol{\xi}, \tau)$.

To numerically solve Equation 27, we use the pseudo-spectral solver from [28]. The initial condition $\boldsymbol{u}_0(\boldsymbol{\xi})$ is sampled from a Gaussian random field: $\boldsymbol{u}_0(\boldsymbol{\xi}) \sim \mathcal{N}\left(0, 7^{3/2}(-\Delta + 49I)^{-5/2}\right)$. Simulations are performed on a 2048×2048 uniform grid. We generate 40 sequences, each simulating 10 seconds of dynamics ($T = 10$). These are downsampled spatially to a 256×256 grid and temporally using a fixed step size $\Delta\tau = 1/32$ s, yielding 320 frames per sequence. Of these, 36 sequences are used for training and the remaining 4 for testing.

For computing the PDE residual $\boldsymbol{\mathcal{R}}(\boldsymbol{u})$, we follow [7, 9], using the discrete Fourier transform to compute spatial derivatives and finite differences for time derivatives. As the proposed diffusion model operates on vorticity over three consecutive frames $[\boldsymbol{u}_{\tau-1}(\boldsymbol{\xi}), \boldsymbol{u}_\tau(\boldsymbol{\xi}), \boldsymbol{u}_{\tau+1}(\boldsymbol{\xi})]$, we approximate the time derivative as: $\partial_\tau \boldsymbol{u}(\boldsymbol{\xi}, \tau) \approx (\boldsymbol{u}_{\tau+1}(\boldsymbol{\xi}) - \boldsymbol{u}_{\tau-1}(\boldsymbol{\xi}))/2\Delta\tau$. The convection and diffusion terms are computed in Fourier space by estimating gradients and Laplacians of the vorticity. The velocity field is derived via the stream function ψ using:

$$\boldsymbol{\nu} = \nabla \times \psi, \quad -\nabla^2 \psi = \boldsymbol{u}, \tag{28}$$

and transformed back to physical space using the inverse Fourier transform.

A.2.4 Burgers' Equation

We study the one-dimensional viscous Burgers' equation with periodic boundary conditions on a unit-length spatial domain $\Omega = (0, 1)$. The governing equation is:

$$\begin{aligned} \partial_\tau \boldsymbol{u}(\xi, \tau) + \partial_\xi \left(\frac{1}{2} \boldsymbol{u}^2(\xi, \tau) \right) = \nu\, \partial_{\xi\xi} \boldsymbol{u}(\xi, \tau), \quad \xi \in \Omega,\ \tau \in (0, T], \\ \boldsymbol{u}(\xi, 0) = \boldsymbol{u}_0(\xi), \quad \xi \in \Omega. \end{aligned} \tag{29}$$

The viscosity is set to $\nu = 0.01$. Following the setup in [8, 28], the initial condition $\boldsymbol{u}_0$ is sampled from a Gaussian random field: $\boldsymbol{u}_0 \sim \mathcal{N}(0,\, 625(\Delta + 25\boldsymbol{I})^{-2})$. We simulate the system for 1 second

using a spectral solver, discretizing both space and time. The spatial grid consists of 128 points, and we take 127 time steps after the initial state, resulting in a solution tensor $\boldsymbol{u}_{0:T} \in \mathbb{R}^{128\times 128}$.

Because the solution is modeled densely over time, the PDE residual can be reliably approximated using finite difference schemes:

$$\boldsymbol{\mathcal{R}}(\boldsymbol{u}) = \partial_\tau \boldsymbol{u}(\xi,\tau) + \partial_\xi \left(\frac{1}{2}\boldsymbol{u}^2(\xi,\tau)\right) - \nu\, \partial_{\xi\xi}\boldsymbol{u}(\xi,\tau). \tag{30}$$

This residual is evaluated at each point in the spatiotemporal domain to assess physical consistency.

A.3 Related work

Reward fine-tuning [16] has been widely applied to adapt pre-trained text-to-image diffusion models to downstream reward functions, such as human preference alignment. Four major classes of paradigms have emerged in this space: (i) *Reward-weighted regression* [29, 30] collects samples and their associated rewards, and then fine-tunes the diffusion model using a reward-weighted loss; (ii) *Policy gradient* methods [21, 17, 31] optimize the expected reward directly using reinforcement learning techniques such as proximal policy optimization [32]; (iii) *Reward backpropagation* [18, 26, 33] assumes a differentiable reward function and directly propagates the reward signal through the diffusion process to update model parameters. This approach is training-efficient but suffers from challenges such as the depth-memory dilemma [26] and reward hacking; (iv) *Value-weighted sampling* [16, 33] avoids parameter updates by using an estimated value function to guide sampling, limited by the quality of the value function approximation.

In this work, we extend reward fine-tuning to physics-informed generation. We focus on the reward backpropagation approach and further investigates a layer-wise truncation schedule beyond conventional step-wise truncation. We also re-examine the necessity of distillation-based regularization in the new context, and replace it with a more efficient yet effective weight regularization.

Physics-informed diffusion models aim to incorporate physical constraints PDEs into diffusion models [8, 7, 15, 9, 13]. *Guidance-based* methods typically build on the DPS framework [19], where physics-based guidance is applied during sampling using point estimates of the data distribution via Tweedie's formula [34]. *Training-based* approaches, on the other hand, integrate physics constraints directly into the learning objective. The specific approaches are discussed in subsection 3.2, with further details provided in subsection A.1. PalSB [9], most closely related to ours, applies DRaFT [18] in a diffusion bridge to improve physical fidelity. However, PalSB is tailored to field reconstruction tasks and the broader potential of reward fine-tuning for physics alignment remains underexplored.

Our method advances this line of work in three key ways. First, we reinterpret existing methods from the reward optmization perspective and identify the key bottleneck from value approximation. Second, compared to PalSB, we investigate a layer-wise truncation strategy. Third, we show that data- or distillation-based regularization in PalSB is unnecessary and propose a more efficient weight-based regularization scheme.

Diffusion Models for PDEs Beyond the core studies discussed earlier, a growing body of work explores the use of diffusion models in the context of PDEs, primarily targeting PDE-solving tasks, with a particular emphasis on fluid dynamics and other time-evolving systems. For example, Kohl et al. [35] introduced an autoregressive formulation for PDE simulation, while Yang and Sommer [36] proposed directly predicting future states. Cachay et al. [37] recently unified denoising time and physical time to improve scalability. Lippe et al. [38] built upon neural operators with iterative denoising refinement to enhance the modeling of high-frequency components and enable long-term rollouts. Other works focus on downstream tasks such as data assimilation and super-resolution. Shu et al. [7] and Jacobsen et al. [13] enforce PDE constraints during inference to improve physical consistency. Huang et al. [8] used an amortized diffusion model combined with inpainting [19] to perform data assimilation on weather datasets. ECI [39] proposed a training- and gradient-free framework for adapting pre-trained functional flow matching (FFM) models [40] to exactly satisfy boundary or initial conditions, rather than full PDE constraints. Rozet and Louppe [41] decomposed long trajectories into short segments with local score estimation to improve memory efficiency, and Qu et al. [42] extended this approach using latent diffusion models for data assimilation on ERA5 weather datasets. Ruhe et al. [43] further introduced a frame-dependent noising process for video and

fluid dynamics generation using local scores. DiffPhyCon [44] addresses PDE control problems by training diffusion models over both state trajectories and control sequences, guided by the control objective. Shysheya et al. [45] conducted a comprehensive comparison of conditional diffusion models for PDE simulation tasks.

While these works share some overlap with ours in enforcing physical constraints, their primary focus lies in PDE solving, where the objective is to achieve high reconstruction or solution accuracy. In contrast, our approach centers on enforcing physical consistency intrinsically within the generative model. That is, we focus on the model's inherent ability to produce samples that satisfy physical laws. Furthermore, our method can be naturally extended to conditional settings by adapting the generative model to incorporate conditioning inputs.

A.4 Limitations and Future Directions

In this work, we focus on the unconditional generation setting. However, PIRF can be readily extended to task-specific conditional settings, such as field reconstruction [9]. A more challenging direction is adapting a fine-tuned unconditional model to support a variety of conditioning types under few-step sampling—for example, solving forward or inverse PDE problems with sparse observations [8]. In such cases, combining reward fine-tuning with additional control mechanisms [46, 47] may offer a promising solution. Moreover, our current implementation uses deterministic sampling for fine-tuning. Exploring stochastic samplers may further improve performance and generalization. Finally, like prior works, we assume the physics-based reward is fully known and differentiable. Extending PIRF to settings with partially known or non-differentiable physics remains an exciting future direction, where policy gradient-based approaches [21, 17, 31] could offer a viable alternative.