Ultra Fast PDE Solving via Physics Guided Few-step Diffusion

Cindy Xiangrui Kong[*1], **Yueqi Wang**[*1], **Haoyang Zheng**[1], **Weijian Luo**[2], **Guang Lin**[1]
[1]Purdue University
[2] Independent Researcher
{kong146, wang7406, zheng528, guanglin}@purdue.edu
pkulwj1994@icloud.com
[*]Equal contribution.

Abstract

Diffusion-based models have demonstrated impressive accuracy and generalization in solving partial differential equations (PDEs). However, their practical utility still suffers from high sampling costs and insufficient physical consistency, mainly due to many-step iterative sampling and insufficient physics guidance. To address these issues, we propose *Phys-Instruct*, a physics-guided distillation framework that compresses a pre-trained diffusion PDE solver into a few-step generator while explicitly enforcing PDE consistency through distillation guidance. We reformulate the training objective as a constrained problem with PDE residual, and derive a score-function identity that leads to a practical physics-guided generator update. Across five PDE benchmarks, Phys-Instruct achieves orders-of-magnitude faster inference while reducing the PDE Error by factors of 2.4 to 15.5, compared with the corresponding 200-step EDM teachers. Moreover, the resulting unconditional student model functions as a compact prior, enabling efficient and physically consistent inference for various downstream conditional tasks. Our results indicate that Phys-Instruct is an effective and efficient framework for ultra-fast PDE solving powered by generative models.

1 Introduction

Diffusion models have recently shown strong empirical performance for modeling spatially structured scientific fields, and have been increasingly explored as flexible priors for *Partial Differential Equation (PDE)* related data [7, 69]. In PDE solving problems, such priors are appealing not only for unconditional field generation, but also for conditional tasks such as forward prediction [39], inverse problems [5, 67], and reconstruction from partial observations [16]. However, making diffusion models practical for PDE tasks requires addressing two coupled issues: sampling efficiency and physical consistency.

Sampling Efficiency: High-fidelity samples typically require hundreds to thousands of denoising steps, resulting in large numbers of function evaluations (NFEs) and high sampling cost [46, 14]. For example, DiffusionPDE [16] addresses forward, inverse, and reconstruction PDE problems from sparse observations by running a long reverse diffusion tra-

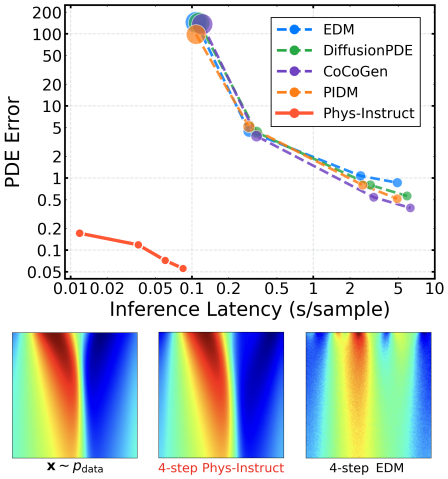


Figure 1: Burgers' generation comparison. **Top:** PDE Error vs. latency. Baselines use 4/10/100/200 steps; Phys-Instruct uses 1–4 steps. **Bottom:** Burgers' samples (left to right): data, Phys-Instruct, and EDM teacher.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

jectory on the order of thousands of steps; Yang and Sommer [57] predicts nonlinear fluid fields with a denoising diffusion model that samples through hundreds of iterative Langevin updates. These approaches demonstrate the potential of diffusion-based PDE modeling, but the required number of function evaluations makes inference expensive and limits their practicality in real-world settings.
Physical Consistency: Standard diffusion training is primarily data-driven and does not explicitly enforce PDE governing equations; as a result, samples can be visually plausible yet exhibit non-negligible PDE error [4]. Moreover, under aggressive acceleration or data-free distillation, small distribution-matching errors can translate into noticeable physics violations, motivating explicit PDE-aware objectives [64].

Existing physics-guided remedies tend to trade one challenge for the other: Some impose physics during inference via guidance or iterative correction, which can improve physical consistency but comes at the cost of higher sampling latency [16]. Others incorporate physics guidance during diffusion training by penalizing PDE error along multi-step denoising trajectories, but sampling typically remains iterative, and the additional constraints can introduce nontrivial trade-offs between sample quality and physical fidelity [4, 2]. In contrast, distillation and other aggressive acceleration schemes reduce sampling steps [31, 47, 26, 40], but physical consistency may degrade without explicit PDE equation constraint. This is particularly pronounced when the budget is limited, where fewer sampling steps imply more aggressive updates along the sampling trajectory, increasing approximation errors when physical constraints are enforced during inference. Together, these highlight a speed-physics trade-off in diffusion-based PDE generation.

To break the fundamental trade-off between sampling efficiency and physical consistency, we introduce **Phys-Instruct**, a principled physics-guided distillation framework. Unlike standard distillation methods [47, 60, 31] that solely focus on distribution matching, Phys-Instruct reformulates distillation as a constrained optimization problem. It compresses a multi-step diffusion teacher into a flexible few-step generator while rigorously enforcing physical laws via a differentiable PDE-error penalty. Crucially, this explicit physics injection occurs entirely during training and requires no paired ground-truth solutions. At inference time, the distilled student model generates highly consistent PDE fields in a fixed few-step forward pass, completely bypassing the need for expensive test-time guidance, iterative corrections, or post-hoc refinements. Extensive evaluations across five diverse PDE benchmarks demonstrate the superiority of Phys-Instruct. It achieves a remarkable dual victory: accelerating inference by orders of magnitude while reducing the PDE error by more than $8\times$ compared to state-of-the-art multi-step diffusion baselines. Furthermore, the distilled unconditional generator serves as a highly compact and versatile diffusion prior. By seamlessly integrating it with a ControlNet-style conditional branch [62], Phys-Instruct can be repurposed for various challenging downstream tasks—including forward prediction, inverse problems, and partial observation reconstructions—enabling accurate, physically faithful solving with unprecedented efficiency.

Our **main contributions** are three-fold: (i) we propose Phys-Instruct, a training-time physics-guided distillation framework that compresses a multi-step diffusion teacher into a one-/few-step generator for PDEs with no test-time physics guidance or control steps; (ii) we demonstrate that incorporating PDE error as a constraint improves distillation over the no physics guidance variant, while requiring no paired ground-truth training data supervision during distillation; (iii) We show the distilled unconditional generator serves as a reusable diffusion prior for downstream conditional PDE tasks (forward/inverse problem with full/partial observations), with favorable accuracy–latency trade-offs.

2 Related Work

Diffusion Priors for PDE Fields Diffusion models have emerged as powerful tools for modeling unconditional PDE distributions [61] and solving conditional tasks such as forward prediction, inverse problems, and partial observation reconstruction [43, 3, 36]. However, rigorously incorporating governing physical laws remains a fundamental challenge. Existing approaches are either purely data-driven [8, 50] or attempt to explicitly inject physics through inference-time guidance [16, 58, 18] or training-time regularization [4, 42]. However, these physics-aware paradigms suffer from an inherent speed-physics trade-off: inference-time guidance severely exacerbates the already sluggish multi-step sampling process, whereas training-time regularization still relies on iterative sampling and struggles to maintain physical fidelity under highly constrained computational budgets. To break this dilemma, our work establishes a novel paradigm by incorporating physics explicitly and exclusively at *distillation time*. By unifying distribution matching with physics-informed learning, we

distill a multi-step teacher into a highly efficient one-/few-step generator, ensuring rigorous physical consistency without relying on expensive test-time physics guidance or post-hoc iterative corrections.

Few-Step Diffusion and Distillation. Diffusion sampling is typically implemented by iteratively solving reverse-time dynamics (e.g., via SDE/ODE solvers), where high-fidelity generation often requires hundreds of computationally expensive function evaluations [46, 19]. To alleviate this burden, a broad line of work accelerates sampling either through advanced training-free numerical solvers [28, 10, 71, 38, 41] or by distilling multi-step teachers into efficient one-/few-step generators [40, 26]. Within the distillation paradigm, prominent approaches include consistency models, which enforce trajectory consistency across noise levels [47, 44, 27, 13, 30, 20, 12], and distribution-matching objectives rooted in Diff-Instruct [31], which have inspired a wide array of fast-generation extensions [51, 33, 32, 56, 60, 59, 55, 11, 70, 17, 34, 35, 68, 54]. While well-established in image and text generation, extending diffusion distillation to PDE fields requires preserving physical consistency alongside distributional fidelity. Recent efforts [64], addresses this challenge by incorporating PDE residual penalties on generated samples during distillation. Phys-Instruct approaches this problem through a penalized integral KL formulation that combines teacher–student distribution matching with differentiable PDE constraints. This formulation supports flexible one-/few-step generation without inference-time physics guidance or iterative correction, while yielding a reusable generative prior for downstream conditional PDE tasks.

3 Problem Formulation and Diffusion Preliminaries

3.1 PDE Field Generation Setup

We consider PDE problems on a domain $\mathcal{D}$, where $\mathcal{D} = \Omega_x \subset \mathbb{R}^d$ for static systems and $\mathcal{D} = \Omega_x \times \Omega_t$ for dynamic systems. Let $\xi \in \mathcal{D}$ denote the coordinates. We collect all physical fields into $\mathbf{x}(\xi) \in \mathbb{R}^c$ and partition it as $\mathbf{x}(\xi) = \left[\mathbf{a}(\xi)^\top, \mathbf{u}(\xi)^\top\right]^\top$, where $\mathbf{a}(\xi) \in \mathbb{R}^{c_1}$, $\mathbf{u}(\xi) \in \mathbb{R}^{c_2}$, and $c = c_1 + c_2$. The pair $(\mathbf{a}, \mathbf{u})$ specifies a PDE instance and its associated solution: in static problems, $\mathbf{a}$ represents coefficients and $\mathbf{u}$ is the corresponding solution field; in dynamic problems, $\mathbf{a}$ specifies the initial condition at $t = 0$ and $\mathbf{u} := \mathbf{u}(\cdot, T)$ denotes the solution state at a target time T. The governing laws, together with boundary/initial constraints, are written as

$$\mathcal{G}[\mathbf{x}](\xi) = 0, \qquad \xi \in \Omega, \tag{1}$$

where $\mathcal{G}$ is a differential operator. Our goal is to generate $\mathbf{x} = (\mathbf{a}, \mathbf{u})$ from the generator g_θ that satisfy (1).

3.2 Diffusion Models

Assume we observe data from an unknown distribution $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$. Diffusion models construct a forward noising process that transforms the initial distribution $p_0 = p_{\text{data}}$ into a simple noise distribution. Specifically, the forward process $\{\mathbf{x}_t\}_{t\in[0,T]}$ is described by the Itô SDE:

$$d\mathbf{x}_t = \mathbf{F}(\mathbf{x}_t, t)\, dt + G(t)\, d\mathbf{w}_t, \tag{2}$$

where $\mathbf{F}$ and $G(t)$ are pre-defined drift and diffusion coefficients, and $\mathbf{w}_t$ is a standard Wiener process. We write $p_t(\mathbf{x}_t)$ for the marginal density of $\mathbf{x}_t$ and $p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$ for the forward transition density induced by (2).

To reverse the noising dynamics, score-based diffusion models estimate the time-dependent score $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$. Since it is intractable in high dimensions, we train a time-conditioned network $s_\phi(\mathbf{x}_t, t)$ to approximate it across noise levels. In practice, s_ϕ is implemented as a discrete noise-level network [45, 14] or a continuous-time network [19, 46]. Training is performed via minimizing a weighted denoising score matching objective [46]:

$$\mathcal{L}_{\text{DSM}}(\phi) = \int_0^T w(t)\, \mathbb{E}_{\mathbf{x}_0, \mathbf{x}_t \mid \mathbf{x}_0} \left\| s_\phi(\mathbf{x}_t, t) - \nabla_{\mathbf{x}_t} \log p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0) \right\|_2^2 \mathrm{d}t, \tag{3}$$

where $\mathbf{x}_0 \sim p_0$, and $\mathbf{x}_t \mid \mathbf{x}_0 \sim p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$, the weighting function $w(t)$ balances the contributions from different noise levels.

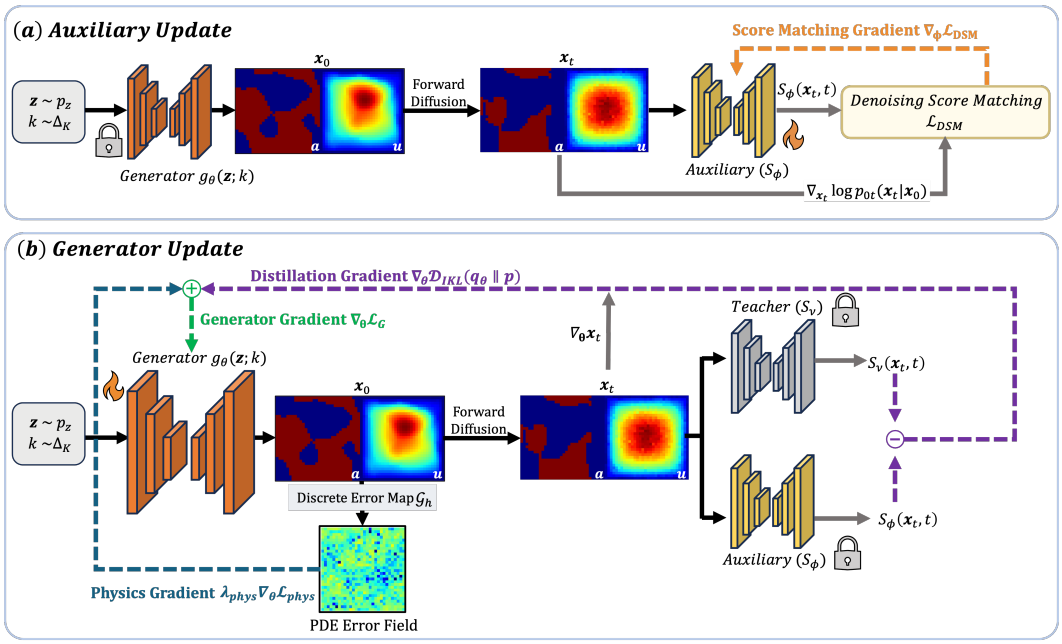


Figure 2: **Overview of Phys-Instruct distillation.** Training alternates between two phases with sampled latent variables $\mathbf{z} \sim p_z$ and step budgets $k \sim \Delta_K$. **(a) Auxiliary update.** With the generator g_θ held fixed, generated samples $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$ are detached and forward-diffused to $\mathbf{x}_t$. The auxiliary network s_ϕ receives $(\mathbf{x}_t, t)$ and is updated by denoising score matching using the target $\nabla_{\mathbf{x}_t} \log p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$. **(b) Generator update.** The frozen teacher s_ν and the auxiliary network s_ϕ, held fixed during this phase, evaluate the same noisy input $(\mathbf{x}_t, t)$. Their score difference is treated as a stop-gradient signal to compute the IKL distillation gradient. This gradient is combined with $\lambda_{\text{phys}} \nabla_\theta \mathcal{L}_{\text{phys}}$ to update g_θ, with the physics guidance evaluated on the clean generated sample $\mathbf{x}_0$.

3.3 Integral Kullback-Leibler Divergence for Diffusion Distillation

Reverse (2) via iterative sampling typically requires a large number of sampling steps, leading to high inference latency, which can limit the practicality of diffusion models for PDE solving. This bottleneck can be alleviated by casting accelerated generation as a distillation problem: a multi-step teacher provides high-quality generative behavior, while a student is distilled from the teacher to produce comparable outputs in substantially fewer function evaluations. To encourage the student to learn the teacher's evolution over time rather than only matching the final output, we adopt a distributional distillation framework that aligns their time-indexed marginals along the diffusion process.

Let $\{p_t(\mathbf{x}_t)\}_{t\in[0,T]}$ denote the teacher marginals induced by the forward diffusion (2), and let the teacher score be given by a frozen network $s_\nu(\mathbf{x}_t, t) \approx \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$. The student model induces a distribution $q_{\theta,0}$ over clean samples $\mathbf{x}_0$ (e.g., through a few-step generator), and we define its time-t marginals $\{q_{\theta,t}(\mathbf{x}_t)\}_{t\in[0,T]}$ by applying the same forward diffusion (2) to $\mathbf{x}_0 \sim q_{\theta,0}$, i.e., $\mathbf{x}_t \mid \mathbf{x}_0 \sim p_{0t}(\mathbf{x}_t \mid \mathbf{x}_0)$. Integral Kullback-Leibler (IKL) divergence generalizes the standard KL divergence by integrating the discrepancy between teacher and student marginals across all diffusion times, which was initially used in Luo et al. [31]:

$$\mathcal{D}_{\text{IKL}}^{[0,T]}(q_\theta \| p) := \int_0^T w(t)\, \mathcal{D}_{\text{KL}}\big(q_{\theta,t} \,\|\, p_t\big)\, \mathrm{d}t = \int_0^T w(t)\, \mathbb{E}_{\mathbf{x}_t \sim q_{\theta,t}} \left[\log \frac{q_{\theta,t}(\mathbf{x}_t)}{p_t(\mathbf{x}_t)}\right] \mathrm{d}t, \tag{4}$$

where $w(t) > 0$ weights the relative importance of different noise levels. Minimizing $\mathcal{D}_{\text{IKL}}^{[0,T]}$ promotes trajectory-level consistency between the student and teacher models is essential for high-quality sampling with limited budgets.

4 A Unified Framework for Physics-Guided Few-Step Distillation

While the IKL objective in (4) provides a principled way to distill a multi-step diffusion teacher into a few-step student, matching the teacher distribution alone does not ensure that the student outputs satisfy the target PDE constraints. Existing physics-guided remedies often enforce feasibility at inference time through guidance or iterative correction, which increases the number of function evaluations and erodes the latency gains of few-step generation.

We therefore propose **Phys-Instruct**, a diffusion distillation framework that *instructs* a few-step student with physics guidance during training. Concretely, we distill a pretrained multi-step diffusion teacher into a *few-step* generator while simultaneously enforcing the PDE constraint (1) through a differentiable PDE error computed on generated samples. In this way, physics enters as explicit training-time supervision for the student, rather than as test-time guidance. As a result, Phys-Instruct produces PDE-consistent samples with a few-step sampler and requires no paired ground-truth solutions during distillation. Figure 2 provides a schematic overview of the proposed framework.

4.1 Flexible Few-Step Sampling with Phys-Instruct

Luo et al. [31] distills a pretrained diffusion teacher into an *implicit one-step* sampler, mapping a Gaussian latent $\mathbf{z}$ to a sample in a single network evaluation. While maximally fast, a single step can be overly restrictive for approximating the teacher's multi-step sampling trajectory. Phys-Instruct generalizes this idea by enabling the student sampler to unroll a *small* number of sampling steps, yielding a distilled diffusion model which can take on step budget $k \in \{1, \dots, K\}$. This provides a simple knob to trade sampling quality for compute while preserving teacher-aligned sampling semantics. Specifically, we parameterize the student generator as a diffusion model g_θ and denote its output with k unrolled steps by $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$, where the latent variable $\mathbf{z} \sim p_z$.

Unified k-Step Formulation. For a given step budget k and fixed noise level σ_{init}, we select $k+1$ noise levels from a fixed sampling schedule, $\sigma_k > \sigma_{k-1} > \cdots > \sigma_0$, where $\sigma_k = \sigma_{\text{init}}$, and draw the initial state $\mathbf{x}_k := \mathbf{z}$, with $\mathbf{z} \sim \mathcal{N}\big(\mathbf{0}, \sigma_{\text{init}}^2 \mathbf{I}\big)$. We then iteratively apply a student-aligned step operator for k steps, $\mathbf{x}_{i-1} = \mathsf{Step}(\mathbf{x}_i; \sigma_i, \sigma_{i-1}, g_\theta)$, $i = k, k-1, \dots, 1$, where $\mathsf{Step}(\cdot)$ belongs to the sampler family that the student diffusion model structure supports (e.g., the Euler [1] or Heun [19] sampler), and is parameterized by the student model g_θ.

4.2 PDE Error and Physics Loss

In practice, all physical fields are defined on a fixed discretized grid $\mathcal{D}_h = \{\xi_i\}_{i=1}^N \subset \Omega$. We approximate the continuous constraint $\mathcal{G}$ in (1) on this grid with a differentiable *discrete PDE error map* $\mathcal{G}_h$, implemented using finite-difference stencils or convolutional filters. The resulting *PDE error (residual) field* is

$$r(\mathbf{x})_i := \mathcal{G}_h[\mathbf{x}](\xi_i), \qquad i = 1, \dots, N, \tag{5}$$

and we summarize it into a scalar physics error via the mean squared PDE error

$$\mathcal{R}(\mathbf{x}) := \frac{1}{N} \sum_{i=1}^{N} \big\| r(\mathbf{x})_i \big\|_2^2. \tag{6}$$

When boundary conditions are prescribed, we impose them explicitly and compute the error only over interior (unconstrained) nodes, which reduces spurious errors introduced by boundary discretization.

4.3 Distillation Objective with Physical Constraints

A central challenge in physics-guided generative modeling is relating the generator's update rule to a rigorous distribution-matching objective while simultaneously adhering to physical laws. Motivated by recent advances that successfully incorporate hard boundaries and explicit constraints into complex sampling and non-convex optimization tasks [63, 22], we propose to reformulate the standard distillation process as a formal constrained optimization problem. Specifically, our goal is to minimize the IKL divergence between the student and teacher distributions, subject to a physical feasibility constraint enforced by the PDE error.

Starting from IKL minimization in (4), we incorporate the physics guidance via (6). Conceptually, this corresponds to a *constrained* distillation problem that matches the teacher distribution while enforcing physical feasibility[1] [25, 9]:

$$\min_{\theta} \ \mathcal{D}_{\mathrm{IKL}}(q_\theta \,\|\, p) \quad \text{s.t.} \quad \mathbb{E}_{\mathbf{x}_0 \sim q_\theta}[\mathcal{R}(\mathbf{x}_0)] \le \varepsilon, \tag{7}$$

where $\mathcal{R}(\mathbf{x}_0) \ge 0$ denotes the PDE error evaluated on a generated sample $\mathbf{x}_0$ by the student, $\varepsilon > 0$ is some small threshold. In our PDE setting, each $\mathbf{x}_0$ encodes the solution field together with the coefficient/forcing fields, and $\mathcal{R}(\mathbf{x}_0)$ is computed with (6).

Building on the unified k-step formulation in Sec. 4.1, we train a *single* student diffusion model that supports flexible sampling budgets $k \in \{1, \dots, K\}$ by randomly unrolling k steps during training. Concretely, we draw

$$k \sim \Delta_K \quad \text{over } \{1, \dots, K\},$$

where Δ_K is a categorical distribution over step budgets, and generate a student sample via the k-step generator $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$ with $\mathbf{z} \sim p_z$.

We define the physics guidance $\mathcal{L}_{\mathrm{phys}}$ on student samples:

$$\mathcal{L}_{\mathrm{phys}}(\theta) := \mathbb{E}_{\mathbf{z} \sim p_z,\, k \sim \Delta_K, \mathbf{x}_0 = g_\theta(\mathbf{z};k)} \Big[\mathcal{R}\big(\mathbf{x}_0\big) \Big]. \tag{8}$$

This term is applied *only* when updating the student generator: the teacher score network and the auxiliary score model are not directly regularized by $\mathcal{L}_{\mathrm{phys}}$, preserving the original IKL distillation dynamics. It is worth noting that we evaluate the PDE error on generated final sample $\mathbf{x}_0$, rather than on the mean prediction such as $\mathbb{E}[\mathbf{x}_0 \mid \mathbf{z}]$. This distinction is critical for nonlinear PDEs: optimizing the mean incurs a Jensen gap that conflicts with distribution matching [64]. Penalizing the PDE error on individual samples directly helps avoid this bias and ensures that generated samples adhere to physical laws.

However, directly solving constrained optimization (7) in high-dimensional deep generative models is computationally intractable. To overcome this, we adopt a penalty relaxation strategy, translating the constrained formulation into a penalized IKL objective where the original distillation is regularized by the physics guidance term in (8). The following theorem formalizes this penalized objective and provides a score-function identity that ensures tractable optimization. Crucially, it demonstrates that the gradient of our objective elegantly decomposes into two distinct components: an IKL distribution-matching term in the spirit of Diff-Instruct, and a physics-informed gradient term that explicitly steers the generator toward PDE satisfaction.

Theorem 4.1 (Physics-Guided Score-Function Identity)**.** *Consider the physics-constrained IKL matching problem* (7)*. As a practical surrogate, we apply a Lagrangian relaxation of* (7) *and define the generator objective, treating $\lambda_{\mathrm{phys}} > 0$ as a trade-off hyperparameter:*

$$\mathcal{L}_G(\theta) := \mathcal{D}_{\mathrm{IKL}}(q_\theta \| p) + \lambda_{\mathrm{phys}}\, \mathcal{L}_{\mathrm{phys}}(\theta), \tag{9}$$

where $\mathcal{L}_{\mathrm{phys}}(\theta)$ is given by (8)*. Under mild regularity conditions stated in Appendix B, the gradient of the implemented generator objective $\mathcal{L}_G(\theta)$ admits the identity*

$$\begin{aligned} \nabla_\theta \mathcal{L}_G(\theta) = \int_0^T w(t)\, \mathbb{E}_{\substack{\mathbf{z} \sim p_{\mathbf{z}},\, k \sim \Delta_K \\ \mathbf{x}_0 = g_\theta(\mathbf{z};k),\, \mathbf{x}_t \sim p_{0t}(\cdot \mid \mathbf{x}_0)}} \Big[\mathrm{SG}\{ s_\phi(\mathbf{x}_t, t) - s_\nu(\mathbf{x}_t, t) \}^\top \nabla_\theta \mathbf{x}_t \Big] \, \mathrm{d}t \\ + \lambda_{\mathrm{phys}}\, \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}},\, k \sim \Delta_K} \Big[\big(\nabla_{\mathbf{x}_0} \mathcal{R}(\mathbf{x}_0) \big)^\top \nabla_\theta g_\theta(\mathbf{z}; k) \Big]_{\mathbf{x}_0 = g_\theta(\mathbf{z};k)}, \end{aligned} \tag{10}$$

where $\mathrm{SG}(\cdot)$ denotes the stop-gradient operator, and $s_\phi(\mathbf{x}_t, t)$ is an auxiliary model trained to approximate the marginal score $\nabla_{\mathbf{x}_t} \log q_{\theta,t}(\mathbf{x}_t)$ of the student-induced distribution at time t. Consequently, the gradient update implemented by (10) *yields a stochastic estimator of the gradient of the penalized IKL objective with PDE guidance.*

We defer the detailed proof to Appendix B for interested readers.

Remark 4.2 (Amortized physics correction)*.* The second term in (10) provides an explicit physics correction direction for the student generator. Indeed, for a fixed $(\mathbf{z}, k)$, the chain rule gives

$$\nabla_\theta \mathcal{R}\big(g_\theta(\mathbf{z}; k)\big) = \Big(\nabla_{\mathbf{x}_0} \mathcal{R}(\mathbf{x}_0) \Big)^\top \nabla_\theta g_\theta(\mathbf{z}; k) \Big|_{\mathbf{x}_0 = g_\theta(\mathbf{z};k)}.$$

[1] We rewrite $\mathcal{D}_{\mathrm{IKL}}^{[0,T]}(\cdot \| \cdot)$ as $\mathcal{D}_{\mathrm{IKL}}(\cdot \| \cdot)$ unless stated otherwise.

Thus, the physics-guided component in (10) pushes the generator outputs along $-\nabla_{\mathbf{x}_0}\mathcal{R}(\mathbf{x}_0)$, i.e., toward lower-residual samples. This can be viewed as amortizing a physics feasibility-improving correction into the student during training, thereby reducing the reliance on additional test-time guidance when operating at small step budgets.

4.4 Phys-Instruct Training Procedure

The Phys-Instruct distillation procedure assumes a pretrained diffusion model s_ν is given. The teacher diffusion models are trained with PDE data to reverse (2) that progressively adds noise to a sample from a real data distribution p_0, where $R(\mathbf{x}_0^*) = 0$ if $\mathbf{x}_0^* \sim p_0$. The teacher diffusion model s_ν is frozen for the whole training procedure.

Phys-Instruct trains the student generator g_θ through two alternative phases between updating the auxiliary diffusion model s_ϕ, and updating the student generator g_θ, with both g_θ and s_ϕ initialized with the teacher's parameters. The former phase follows the standard diffusion model learning procedure, i.e., minimizing loss function (3), with a slight change that $\mathbf{x}_0$ is not from the training dataset but generated from g_θ. The auxiliary model s_ϕ is updated online to track the score $\nabla_{\mathbf{x}_t} \log q_{\theta,t}(\mathbf{x}_t)$ of the current student marginal at noise level t. The latter phase updates the generator's parameter θ using $\mathcal{L}_G(\theta)$, whose tractable gradient $\nabla_\theta \mathcal{L}_G(\theta)$ is derived as (10). The whole procedure is summarized in Algorithm 1.

5 Experiments

PDE Benchmarks. We conduct experiments on 5 different PDE benchmarks, including Darcy flow, the Poisson equation, the non-bounded Navier-Stokes equation, the Burgers' equation, and the inhomogeneous Helmholtz equation. The dataset generation mainly follow released scripts in [16] and [23]. Different benchmarks are of different resolutions as indicated in Table 1. More details about the PDE benchmarks are deferred to Appendix A.

Training. For all benchmarks, Phys-Instruct utilizes the EDM teacher model [19] and jointly generates both the coefficient field $\mathbf{a}$ and the solution field $\mathbf{u}$, except for the Burgers' equation benchmark, where we store only the space–time field $\mathbf{u}$. Teacher models are trained with 50,000 data points. The teacher models for Darcy flow, Poisson, and non-bounded Navier-Stokes are trained with 4 V100 GPUs for around 6 hours. The teacher models for Burgers' and Helmholtz come from [16]. The step operator follows the deterministic second-order Heun sampler in [19]. The distillation is conducted with max sampling step $K = 4$ with 2 A100 GPUs for around 4 hours. A normalized compute comparison and an update-level cost breakdown are provided in Appendix E.

Baselines and Protocols. We evaluate four diffusion-based baselines using the deterministic second-order Heun sampler following [19]. To assess both generation quality and computational efficiency, we report results at 200 steps representing the baselines' full potential, and at 4 steps matching the budget of our proposed method. Our comparison includes the plain EDM teacher model [19] as a purely data-driven reference without physical constraints. We then consider two methods that enforce physics during sampling: DiffusionPDE [16] augments EDM by injecting physics guidance specifically during the final 20% of the sampling trajectory following diffusion posterior sampling (DPS) [6], with this guidance extending to the last 50% when step=4. Similarly, CoCoGen [18] starts from the DiffusionPDE output and performs an additional ten-step physics-guided refinement, or two extra steps specifically for the four-step setting. Finally, we compare against PIDM [4], which distinguishes itself by incorporating physics guidance directly into the training objective rather than during sampling.

5.1 Unconditional PDE Generation Results

As our primary physics consistency metric, we report *PDE Error*↓, the root-mean-square of the PDE error computed per sample as $\sqrt{\mathcal{R}(\mathbf{x}_0)}$ and then averaged over 1,000 samples; the results are summarized in Table 1. The efficiency trade-off is visualized in Figure 1, and the distributional discrepancy, measured by SWD and MMD, is reported in Figure 3. Additional unconditional-generation results, including sample-level variability and resolution-scaling experiments, are provided in Appendix F.

Table 1: Unconditional generation results on PDE benchmarks. We report PDE Error↓ and the number of sampling steps per sample (Step). $\text{NS}_{\text{unb.}}$ denotes the nonbounded Navier–Stokes benchmark, and the lowest PDE Error for each benchmark is **bolded**.

		PDE Error (↓)				
Method	Step	Darcy Flow	Poisson	$\text{NS}_{\text{unb.}}$	Burgers'	Helmholtz
Resolution		32×32×2	32×32×2	64×64×2	128×128×1	128×128×2
Phys-Instruct	1	3.15×10^{-1}	2.16×10^{-2}	3.15×10^{-2}	1.71×10^{-1}	3.75×10^{-1}
	2	2.18×10^{-1}	5.18×10^{-3}	1.46×10^{-2}	1.18×10^{-1}	3.00×10^{-1}
	3	1.35×10^{-1}	4.79×10^{-3}	3.65×10^{-3}	7.17×10^{-2}	2.10×10^{-1}
	4	$\mathbf{8.49\times10^{-2}}$	$\mathbf{3.99\times10^{-3}}$	$\mathbf{3.21\times10^{-3}}$	$\mathbf{5.54\times10^{-2}}$	$\mathbf{1.46\times10^{-1}}$
EDM [19]	4	196.03	16.72	3.33×10^{-1}	144.23	79.77
DiffusionPDE [16]	4	191.50	16.68	3.30×10^{-1}	141.37	78.97
CoCoGen [18]	4	190.87	16.60	3.24×10^{-1}	135.65	77.34
PIDM [4]	4	101.04	22.64	3.44×10^{-1}	97.76	77.86
EDM [19]	200	5.59×10^{-1}	3.14×10^{-2}	3.60×10^{-2}	8.60×10^{-1}	3.54×10^{-1}
DiffusionPDE [16]	200	4.38×10^{-1}	2.64×10^{-2}	3.23×10^{-2}	5.62×10^{-1}	3.38×10^{-1}
CoCoGen [18]	200	3.31×10^{-1}	1.40×10^{-2}	2.92×10^{-2}	3.84×10^{-1}	3.04×10^{-1}
PIDM [4]	200	5.26×10^{-1}	2.36×10^{-2}	3.54×10^{-2}	5.14×10^{-1}	3.15×10^{-1}

It is shown in Table 1 that Phys-Instruct is particularly effective in the low-step regime. Increasing the sampling steps from 1 to 4 yields consistent improvements in PDE error across all benchmarks. Figure 6 further provides qualitative generation results of Darcy flow with 1-4 step Phys-Instruct. Notably, the 1-step Phys-Instruct already outperforms the best Darcy flow and Burgers' baseline even when the baseline uses 200 sampling steps, and the 4-step Phys-Instruct attains the lowest PDE error overall. In contrast, at the same low sampling budget (Step=4), diffusion baselines often produce samples with substantially larger PDE error. With such aggressive step sizes, the correction evaluation in second-order samplers can become sensitive to model and discretization errors, which may degrade the final sample quality, as illustrated in Figure 1. Phys-Instruct alleviates this issue because the student is distilled with the Heun sampler at fixed noise levels, i.e., it matches the teacher along a specific discrete trajectory. This effectively learns a discrete transport map tailored to the few-step schedule, rather than relying on large-step integration of the teacher's continuous-time dynamics.

The efficiency advantage is further illustrated in Figure 1. Latency measures end-to-end sampling time and includes any physics guidance or refinement applied during sampling. Diffusion baselines exhibit a clear compute–quality trade-off: achieving lower PDE error typically requires substantially more sampling steps, whereas very small step budgets often degrade physics consistency. In contrast, Phys-Instruct attains low PDE error at significantly lower latency. Increasing the student budget from Step=1 to Step=4 yields consistent improvements on Burgers', reducing PDE error from 1.71×10^{-1} to 5.54×10^{-2}, while remaining in the low-latency regime.

Beyond physics consistency, we measure distributional discrepancy to held-out data on Burgers' using SWD and MMD, reported in Figure 3. Few-step Phys-Instruct substantially improves both metrics compared to low-budget teacher sampling. At Step$=4$, SWD and MMD improve by 82% and 63%, respectively. Meanwhile, the four-step Phys-Instruct achieves SWD and MMD that are comparable to those of the high-budget EDM teacher, albeit slightly worse. These results indicate effective distillation and suggest that incorporating physics guidance during training does not induce mode collapse. The remaining gap is expected, since Phys-Instruct is distilled from the teacher under a constrained step budget rather than directly optimizing SWD/MMD to match the data distribution.

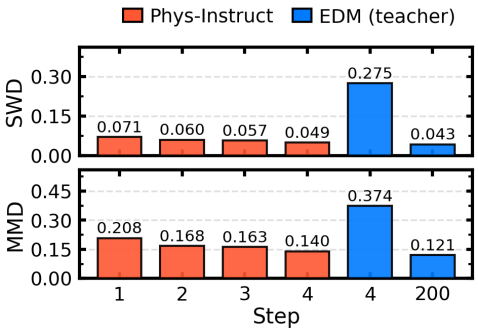


Figure 3: Distributional discrepancy on the Burgers' benchmark measured by SWD and MMD (reported as $\sqrt{\text{MMD}^2}$).

5.2 Guidance Schedule and Strength Ablations

We ablate the role of physics guidance in Phys-Instruct to understand how it improves physical consistency. Table 2 reports results on Darcy flow at resolution 32×32 with fixed $\sigma_{\text{init}} = 1.5$ under one-step sampling, measured by the PDE Error. Removing PDE guidance (w/o Guidance) leads to degradation in physics consistency, with a PDE Error of 8.75 on Darcy flow. In contrast, adding the physics guidance with the default weight ($\lambda_{\text{phys}} = 5\times10^{-3}$) decreases PDE Error by approximately 27 times. Figure 4 further illustrates the training dynamics. The trajectory without physics guidance plateaus at a high PDE Error, whereas applying physics guidance from the start drives a faster and more sustained decrease in PDE Error. When guidance is activated only after the PDE error plateaus, the error decreases after the intervention, but does not fully match the best performance. Overall, these results indicate that PDE guidance provides an effective optimization signal and that introducing it throughout distillation yields the best result.

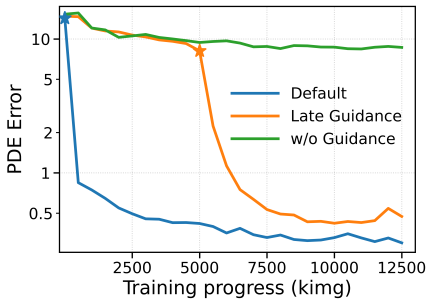


Figure 4: Darcy-flow PDE error during one-step Phys-Instruct distillation under different guidance schedules. $*$ indicate when guidance starts.

Additionally, Table 2 indicates that the method is reasonably robust to the choice of λ_{phys} within a practical range. Overly weak physics weighting noticeably degrades physical consistency, whereas a larger weight performs comparably to the default. Additional sensitivity analyses for λ_{phys} and σ_{init}, as well as practical tuning strategy for λ_{phys} are provided in Appendix G.

5.3 Conditional Downstream Tasks

We further show that the distilled students provide strong *unconditional priors* for downstream PDE tasks. Specifically, we freeze the unconditional Phys-Instruct as a backbone prior and attach a ControlNet-style conditional branch [62] to incorporate paired task data. Due to space constraints, details of the conditional pipeline implementation and baselines are deferred to Appendix H. We showcase the forward and inverse problem on Darcy flow and Helmholtz with resolution 128×128. Extra downstream experiments can be found in Appendix I.

Table 2: Ablation of Physics guidance in one-step Phys-Instruct training on Darcy flow. **Default** denotes full training configuration. We report the PDE Error (↓). Kimg is the starting point of adding physics guidance.

Setting	Guidance	kimg	λ_{phys}	PDE Error↓
Default	✓	0	5e−3	$\mathbf{3.15\times10^{-1}}$
w/o Guidance	✗	–	0	8.75
Late Guidance	✓	5k	5e−3	4.07×10^{-1}
Weak Guidance	✓	0	1e−3	3.59×10^{-1}
Strong Guidance	✓	0	1e−2	3.18×10^{-1}

Table 3 shows that one-step Phys-Instruct consistently outperforms diffusion baselines using 2000 steps on both Darcy flow and Helmholtz. On Darcy flow, it reduces forward prediction error by about 5–9× and inverse error by about 28–38×, while also lowering PDE Error. On Helmholtz, it achieves about 5–7× lower forward error and 4–5× lower inverse error, again with improved PDE consistency. Figure 5 shows qualitative comparison of Phys-Instruct with other baseline methods on Helmholtz equation forward problem. It demonstrates that Phys-Instruct achieves low relative error and PDE error with details well captured. These results show that the distilled prior transfers effectively to conditional PDE tasks at orders-of-magnitude lower sampling cost.

Table 3: Forward/inverse results on Darcy flow and Helmholtz at 128×128. Prediction and PDE errors are reported (↓); Darcy inverse uses absolute error, while others use relative error. Step denotes inference sampling steps.

Method	Step	Darcy Flow				Helmholtz			
		Forward		Inverse		Forward		Inverse	
		Rel. Err.	PDE Err.	Abs. Err.	PDE Err.	Rel. Err.	PDE Err.	Rel. Err.	PDE Err.
Phys-Instruct	1	**0.50%**	**1.08**	**0.35%**	**1.54**	**1.86%**	**0.42**	**4.58%**	**0.31**
DiffusionPDE	2000	2.52%	2.80	9.92%	3.11	9.69%	1.48	17.87%	0.54
CoCoGen	2000	2.55%	2.76	10.11%	3.05	9.70%	1.07	17.86%	0.47
PIDM	2000	4.39%	3.72	13.46%	3.97	13.76%	1.36	22.65%	0.54

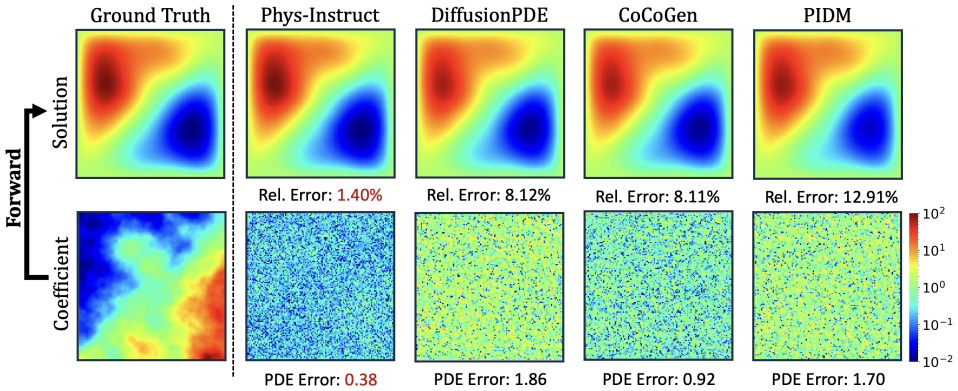


Figure 5: Qualitative comparison for the Helmholtz forward problem (corresponding to Table 3). The leftmost column shows the ground-truth fields, and the remaining columns show predictions from different methods. The top row visualizes the predicted solution u, and the bottom row shows the pointwise absolute PDE error $|r(\mathbf{x})_i|$ over the grid. We annotate each method with the solution relative error and the overall PDE Error↓, highlighting that Phys-Instruct achieves the lowest PDE error and best agreement with the ground truth.

6 Conclusion and Future Work

We propose Phys-Instruct, a physics-guided distillation framework that compresses a diffusion teacher into a few-step generator for PDE solution generation. Phys-Instruct augments IKL-based distillation objective with physics constraints, yielding students that produce physically consistent samples with dramatically fewer sampling steps, requiring no test-time physics correction. Empirically, the few-step Phys-Instruct achieves substantially lower PDE Error than diffusion baselines that require hundreds of sampling steps, and the distilled one-step model serves as a strong unconditional backbone for downstream conditional tasks.

There are several promising directions for future work. First, it would be valuable to combine Phys-Instruct with more advanced distillation techniques, such as SIM [32] and Uni-Instruct [51], to better transfer the knowledge from the teacher while further improving the diversity of the student-generated samples. For conditional tasks, adapters [37] or LoRA [15] may provide parameter-efficient alternatives to the current ControlNet setup. Currently Phys-Instruct is limited to fixed resolution setting, extending the framework to and resolution-robust or resolution-free settings [58, 49] is another important direction. We hope Phys-Instruct will provide a foundation for building efficient physics-aware generative models.

Acknowledgments and Disclosure of Funding

Computational experiments were performed in part using the Gautschi community cluster operated by Purdue IT's Rosen Center for Advanced Computing (RCAC), and the Majda GPU server in the Department of Mathematics at Purdue University.

Guang Lin gratefully acknowledges the support from DARPA MATHBAC program under grant number: HR0011-26-3-E152, the National Science Foundation (DMS-2533878, DMS-2053746, DMS-2134209, ECCS-2328241, CBET-2347401 and OAC-2311848), and U.S. Department of Energy (DOE) Office of Science Advanced Scientific Computing Research program under the "Uncertainty Quantification for Multifidelity Operator Learning (MOLUcQ)" project (Project No. 81739), DE-SC0023161, the SciDAC LEADS Institute, and DOE–Fusion Energy Science, under grant number: DE-SC0024583.

The authors declare no competing interests.

References

[1] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics, Philadelphia,

PA, 1998. ISBN 978-0-89871-412-8. doi: 10.1137/1.9781611971392.

[2] Giacomo Baldan, Qiang Liu, Alberto Guardone, and Nils Thuerey. Flow matching meets pdes: A unified framework for physics-constrained generation. *arXiv preprint arXiv:2506.08604*, 2025. URL https://arxiv.org/abs/2506.08604.

[3] Lorenzo Baldassari, Ali Siahkoohi, Josselin Garnier, Knut Solna, and Maarten V de Hoop. Conditional score-based diffusion models for Bayesian inference in infinite dimensions. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://openreview.net/forum?id=voG6nEW9BV.

[4] Jan-Hendrik Bastek, WaiChing Sun, and Dennis M Kochmann. Physics-informed diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

[5] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-Informed Neural Networks (PINNs) for Fluid Mechanics: A Review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021.

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

[7] Giannis Daras, Hyungjin Chung, Chieh-Hsin Lai, Yuki Mitsufuji, Jong Chul Ye, Peyman Milanfar, Alexandros G. Dimakis, and Mauricio Delbracio. A survey on diffusion models for inverse problems, 2024. URL https://arxiv.org/abs/2410.00083.

[8] Agnimitra Dasgupta, Harisankar Ramaswamy, Javier Murgoitio-Esandi, Ken Foo, Runze Li, Qifa Zhou, Brendan F. Kennedy, and Assad A. Oberai. Conditional score-based diffusion models for solving inverse elasticity problems. *Computer Methods in Applied Mechanics and Engineering*, 433:117425, 2025. doi: 10.1016/j.cma.2024.117425. URL https://www.sciencedirect.com/science/article/pii/S0045782524006807.

[9] Wei Deng, Yu Chen, Nicole Yang, Hengrong Du, Qi Feng, and Ricky T. Q. Chen. Reflected Schrödinger Bridge for Constrained Generative Modeling. *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2024.

[10] Tim Dockhorn, Arash Vahdat, and Karsten Kreis. GENIE: Higher-order denoising diffusion solvers. In *Advances in Neural Information Processing Systems*, volume 35, pages 30150–30166, 2022.

[11] 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 (NeurIPS)*, 36:79858–79885, 2023.

[12] Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean Flows for One-Step Generative Modeling. *arXiv preprint arXiv:2505.13447*, 2025.

[13] Zhengyang Geng, Ashwini Pokle, Weijian Luo, Justin Lin, and J. Zico Kolter. Consistency Models Made Easy. In *Proc. of the International Conference on Learning Representation (ICLR)*. OpenReview.net, 2025.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[16] Jiahe Huang, Guandao Yang, Zichen Wang, and Jeong Joon Park. Diffusionpde: Generative pde-solving under partial observation. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[17] Zemin Huang, Zhengyang Geng, Weijian Luo, and Guo-jun Qi. Flow generator matching. *arXiv preprint arXiv:2410.19310*, 2024.

[18] 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*, 2025. doi: 10.1137/24M1636071.

[19] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022)*, 2022.

[20] Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency Trajectory Models: Learning Probability Flow ODE Trajectory of Diffusion. In *Proc. of the International Conference on Learning Representation (ICLR)*, 2024.

[21] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo K Rohde. Generalized sliced wasserstein distances. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

[22] Andrew Lamperski. Projected Stochastic Gradient Langevin Algorithms for Constrained Sampling and Non-Convex Learning. In *Proc. of Conference on Learning Theory (COLT)*, 2021.

[23] Zongyi Li, Nikola 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. URL `https://openreview.net/forum?id=c8aUC5b6fPi`.

[24] 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/IMS Journal of Data Science*, 1(3):1–27, 2024. doi: 10.1145/3648506.

[25] Xingchao Liu, Xin Tong, and Qiang Liu. Sampling with Trusthworthy Constraints: A Variational Gradient Framework. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:23557–23568, 2021.

[26] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023. URL `https://openreview.net/forum?id=Xc99ZMQtrX`.

[27] Cheng Lu and Yang Song. Simplifying, Stabilizing and Scaling Continuous-time Consistency Models. In *Proc. of the International Conference on Learning Representation (ICLR)*. OpenReview.net, 2025.

[28] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022.

[29] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning Nonlinear Operators via Deeponet Based on the Universal Approximation Theorem of Operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[30] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference. *arXiv preprint arXiv:2310.04378*, 2023.

[31] Weijian Luo, Tianyang Hu, Shifeng Zhang, Jiacheng Sun, Zhenguo Li, and Zhihua Zhang. Diff-instruct: A universal approach for transferring knowledge from pre-trained diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[32] Weijian Luo, Zemin Huang, Zhengyang Geng, J. Zico Kolter, and Guo-jun Qi. One-Step Diffusion Distillation through Score Implicit Matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[33] Weijian Luo, Colin Zhang, Debing Zhang, and Zhengyang Geng. Diff-instruct*: Towards human-preferred one-step text-to-image generative models, 2024. URL https://arxiv.org/abs/2410.20898.

[34] Yihong Luo, Xiaolong Chen, Xinghua Qu, Tianyang Hu, and Jing Tang. You Only Sample Once: Taming One-Step Text-to-Image Synthesis by Self-Cooperative Diffusion GANs. In *Proc. of the International Conference on Learning Representation (ICLR)*. OpenReview.net, 2025.

[35] Yihong Luo, Tianyang Hu, Weijian Luo, Kenji Kawaguchi, and Jing Tang. Reward-Instruct: A Reward-Centric Approach to Fast Photo-Realistic Image Generation. *arXiv preprint arxiv.org/abs/2503.13070*, 2025.

[36] Amirhossein Mollaali, Bongseok Kim, Christian Moya, and Guang Lin. pADAM: A Plug-and-Play All-in-One Diffusion Architecture for Multi-Physics Learning, 2026. URL https://arxiv.org/abs/2603.16757.

[37] Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, Ying Shan, and Xiaohu Qie. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4296–4304, 2024.

[38] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *Proceedings of the 38th International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021.

[39] 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.

[40] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

[41] Andy Shih, Suneel Belkhale, Stefano Ermon, Dorsa Sadigh, and Nima Anari. Parallel sampling of diffusion models. In *Advances in Neural Information Processing Systems*, volume 36, pages 56403–56430, 2023.

[42] 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. doi: 10.1016/j.jcp.2023.111972.

[43] Aliaksandra Shysheya, Cristiana Diaconu, Federico Bergamin, Paris Perdikaris, José Miguel Hernández-Lobato, Richard E Turner, and Emile Mathieu. On conditional diffusion models for PDE simulations. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://openreview.net/forum?id=nQl8EjyMzh.

[44] Yang Song and Prafulla Dhariwal. Improved Techniques for Training Consistency Models. In *Proc. of the International Conference on Learning Representation (ICLR)*. OpenReview.net, 2024.

[45] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, pages 11895–11907, 2019. URL https://proceedings.neurips.cc/paper/2019/file/3001ef257407d5a371a96dcd947c7d93-Paper.pdf.

[46] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of the International Conference on Learning Representations (ICLR) 2021*, 2021.

[47] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 32211–32252. PMLR, 2023. URL `https://proceedings.mlr.press/v202/song23a.html`.

[48] Danica J Sutherland, Hsiao-Yu Tung, Heiko Strathmann, Soumyajit De, Aaditya Ramdas, Alex Smola, and Arthur Gretton. Generative models and model criticism via optimized maximum mean discrepancy. In *International Conference on Learning Representations (ICLR)*, 2017.

[49] Sifan Wang, Zehao Dou, Tong-Rui Liu, and Lu Lu. Fundiff: Diffusion models over function spaces for physics-informed generative modeling, 2025.

[50] Ting Wang, Petr Plecháč, and Jaroslaw Knap. Generative diffusion learning for parametric partial differential equations, 2023. URL `https://arxiv.org/abs/2305.14703`.

[51] Yifei Wang, Weimin Bai, Colin Zhang, Debing Zhang, Weijian Luo, and He Sun. Uni-Instruct: One-step Diffusion Model through Unified Diffusion Divergence Instruction. *arXiv preprint arXiv:2505.20755*, 2025.

[52] Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 53681–53705. PMLR, 2024.

[53] Jiqing Wu, Zhiwu Huang, Dinesh Acharya, Wen Li, Janine Thoma, Danda Pani Paudel, and Luc Van Gool. Sliced wasserstein generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3713–3722, 2019.

[54] Junyi Wu, Weijian Luo, Haoyang Zheng, Ruizhe Zhang, and Guang Lin. Diff-Instruct with Diffused Reward: Towards Principled One-step Generator RL, 2026. URL `https://arxiv.org/abs/2605.24001`.

[55] Sirui Xie, Zhisheng Xiao, Diederik P. Kingma, Tingbo Hou, Ying Nian Wu, Kevin P. Murphy, Tim Salimans, Ben Poole, and Ruiqi Gao. EM Distillation for One-step Diffusion Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[56] Yilun Xu, Weili Nie, and Arash Vahdat. One-step Diffusion Models with f-Divergence Distribution Matching. *arXiv preprint arXiv:2502.15681*, 2025.

[57] Gefan Yang and Stefan Sommer. A denoising diffusion model for fluid field prediction, 2023.

[58] Jiachen Yao, Abbas Mammadov, Julius Berner, Gavin Kerrigan, Jong Chul Ye, Kamyar Azizzadenesheli, and Anima Anandkumar. Guided diffusion sampling on function spaces with applications to pdes, 2025. URL `https://arxiv.org/abs/2505.17004`. arXiv:2505.17004v2.

[59] Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and Bill Freeman. Improved Distribution Matching Distillation for Fast Image Synthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[60] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-Step Diffusion with Distribution Matching Distillation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[61] Mingze Yuan, Pengfei Jin, Na Li, and Quanzheng Li. Pirf: Physics-informed reward fine-tuning for diffusion models, 2025.

[62] 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 (ICCV)*, pages 3836–3847, 2023.

[63] Ruqi Zhang, Qiang Liu, and Xin Tong. Sampling in Constrained Domains with Orthogonal-Space Variational Gradient Descent. *Advances in Neural Information Processing Systems (NeurIPS)*, 35:37108–37120, 2022.

[64] Yi Zhang and Difan Zou. Physics-informed distillation of diffusion models for pde-constrained generation, 2025. URL https://arxiv.org/abs/2505.22391.

[65] Haoyang Zheng and Guang Lin. Muti-Fidelity Prediction and Uncertainty Quantification with Laplace Neural Operators for Parametric Partial Differential Equations. *arXiv preprint arXiv:2502.00550*, 2025.

[66] Haoyang Zheng and Guang Lin. LES-SINDy: Laplace-Enhanced Sparse Identification of Nonlinear Dynamical Systems. *Journal of Computational Physics*, page 114443, 2026.

[67] Haoyang Zheng, Yao Huang, Ziyang Huang, Wenrui Hao, and Guang Lin. HomPINNs: Homotopy Physics-Informed Neural Networks for Solving the Inverse Problems of Nonlinear Differential Equations with Multiple Solutions. *Journal of Computational Physics*, 500:112751, 2024.

[68] Haoyang Zheng, Xinyang Liu, Cindy Xiangrui Kong, Nan Jiang, Zheyuan Hu, Weijian Luo, Wei Deng, and Guang Lin. Ultra-fast language generation via discrete diffusion divergence instruct. *arXiv preprint arXiv:2509.25035*, 2025.

[69] Hongkai Zheng, Wenda Chu, Bingliang Zhang, Zihui Wu, Austin Wang, Berthy T. Feng, Caifeng Zou, Yu Sun, Nikola Kovachki, Zachary E. Ross, Katherine L. Bouman, and Yisong Yue. Inversebench: Benchmarking plug-and-play diffusion priors for inverse problems in physical sciences. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=U3PBITXNG6.

[70] Mingyuan Zhou, Huangjie Zheng, Zhendong Wang, Mingzhang Yin, and Hai Huang. Score Identity Distillation: Exponentially Fast Distillation of Pretrained Diffusion Models for One-Step Generation. In *Proc. of the International Conference on Machine Learning (ICML)*. OpenReview.net, 2024.

[71] Zhenyu Zhou, Defang Chen, Can Wang, and Chun Chen. Fast ODE-based sampling for diffusion models in around 5 steps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7777–7786, 2024.

Contents

1 Introduction **1**

2 Related Work **2**

3 Problem Formulation and Diffusion Preliminaries **3**
3.1 PDE Field Generation Setup . . . 3
3.2 Diffusion Models . . . 3
3.3 Integral Kullback-Leibler Divergence for Diffusion Distillation . . . 4

4 A Unified Framework for Physics-Guided Few-Step Distillation **5**
4.1 Flexible Few-Step Sampling with Phys-Instruct . . . 5
4.2 PDE Error and Physics Loss . . . 5
4.3 Distillation Objective with Physical Constraints . . . 5
4.4 Phys-Instruct Training Procedure . . . 7

5 Experiments **7**
5.1 Unconditional PDE Generation Results . . . 7
5.2 Guidance Schedule and Strength Ablations . . . 9
5.3 Conditional Downstream Tasks . . . 9

6 Conclusion and Future Work **10**

A PDE Benchmark Details **17**
A.1 Darcy Flow . . . 17
A.2 Poisson Equation . . . 18
A.3 Inhomogeneous Helmholtz Equation . . . 18
A.4 Non-bounded Navier–Stokes Equation . . . 18
A.5 Burgers' Equation . . . 19

B Proof of Theorem 4.1 **19**

C Phys-Instruct Algorithm **20**

D Distributional Metrics **21**
D.1 Additional Distributional Coverage and Sample Diversity on Burgers' Equation . . . 21

E Normalized Compute and Update-Level Breakdown **22**

F Additional Phys-Instruct Generation Results **22**
F.1 Sample-Level Variability . . . 22
F.2 Resolution Scalability on Poisson . . . 22
F.3 Qualitative Results of Darcy Flow . . . 23
F.4 Physical Diagnostics Beyond PDE Error . . . 23

G Extra Ablation Studies **26**
G.1 Sensitivity to the Physics Guidance Weight λ_{phys} . . . 26
G.2 Sensitivity to the Initial Noise Level σ_{init} . . . 26
G.3 Ablation on Step Operator . . . 27

H Conditional Downstream Tasks with ControlNet **27**

I Extra Downstream Task Results **29**
I.1 Comparison with Deterministic Surrogate Models . . . 29
I.2 Additional Forward and Inverse Problem . . . 29
I.3 Reconstruction Problem . . . 29

A PDE Benchmark Details

This section provides benchmark details for the five PDEs used in our experiments. A PDE sample is denoted by $\mathbf{x} = (\mathbf{a}, \mathbf{u})$, where $\mathbf{a}$ collects coefficient/forcing/initial-condition fields and $\mathbf{u}$ is the corresponding solution field or the target solution state for dynamic problems.

For each benchmark, all fields are represented on a fixed grid $\mathcal{D}_h = \{\xi_i\}_{i=1}^{N} \subset \Omega$. For static problems on $\Omega_x \subset \mathbb{R}^d$, we take $\xi = \mathbf{c} \in \Omega_x$ and $\mathcal{D}_h = \{\mathbf{c}_i\}_{i=1}^{N_x}$. For dynamic problems on $\Omega = \Omega_x \times \Omega_t$, we use the space–time coordinate $\xi = (\mathbf{c}, \tau)^\top$ and a tensor grid $\mathcal{D}_h = \{(\mathbf{c}_i, \tau_j)\}_{i=1,j=1}^{N_x,N_t}$. Following Huang et al. [16], we generate $50{,}000$ samples as training data and $20{,}000$ as test data per PDE. Darcy flow coefficient fields are drawn from Gaussian random fields (GRFs), and solutions are computed by solving the resulting sparse grid-discretized elliptic system. Following the FNO benchmark [23], we generate the periodic Navier–Stokes data using a Fourier pseudo-spectral solver in the stream-function formulation. Poisson and inhomogeneous Helmholtz data are generated using second-order finite-difference schemes, and Burgers' data follow Zheng and Lin [66, 65] with a spectral solver to produce space–time fields. All fields are stored as discretized arrays on the target grids.

On the same grid $\mathcal{D}_h$, we follow Section 4.2 to compute the benchmark physics penalty: $r(\mathbf{x})_i := \mathcal{G}_h[\mathbf{x}](\xi_i)$ and $\mathcal{R}(\mathbf{x})$ is given by (6). When boundary values are prescribed, we enforce them explicitly and evaluate $\mathcal{R}(\mathbf{x})$ only on interior (unconstrained) nodes to avoid boundary-discretization artifacts. The explicit form of $\mathcal{G}_h$ (and thus $\mathcal{R}(\mathbf{x})$) is benchmark-dependent and will be specified for each PDE below.

The PDE datasets are used to learn the data distribution p_0 by training the multi-step diffusion *teacher* score model s_ν, which is then frozen, and to construct evaluation/conditional test instances. In contrast, student distillation does not require paired ground-truth supervision: during distillation, clean samples are generated on-the-fly as $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$, and physics guidance is applied by evaluating the PDE error penalty $\mathcal{R}(\mathbf{x}_0)$ on these samples.

A.1 Darcy Flow

We consider the static Darcy flow on $\Omega_x = (0,1)^2$ with homogeneous Dirichlet boundary:

$$\begin{cases} -\nabla \cdot \big(a(\mathbf{c})\, \nabla u(\mathbf{c})\big) = q(\mathbf{c}), & \mathbf{c} \in \Omega_x, \\ u(\mathbf{c}) = 0, & \mathbf{c} \in \partial\Omega_x, \end{cases} \tag{11}$$

where the forcing is fixed as $q(\mathbf{c}) \equiv 1$.

Each dataset sample is obtained by first drawing a random permeability field and then solving the PDE. Following DiffusionPDE Huang et al. [16], we construct heterogeneous coefficients from a Gaussian random field sampled on the target grid. Specifically, we draw a zero-mean GRF on a 32×32 or 128×128 grid using a Gaussian-correlated spectral sampler, and define a binary-valued permeability by pointwise thresholding:

$$\mathbf{a}_{ij} = \begin{cases} 12, & \mathbf{m}_{ij} \geq 0, \\ 3, & \mathbf{m}_{ij} < 0, \end{cases} \qquad (i,j) \in \mathcal{D}_h,$$

where $\mathbf{m} \in \mathbb{R}^{32\times 32}$ or $\mathbb{R}^{128\times 128}$ denotes the sampled GRF and $\mathcal{D}_h$ is the uniform grid over Ω_x. Given $\mathbf{a}$, we solve the corresponding discretized Darcy system with homogeneous Dirichlet boundary conditions, and store the resulting fields on the same grid.

We denote the discretized fields by boldface symbols: $\mathbf{a}$ and $\mathbf{u}$ are the grid-sampled permeability and solution fields, respectively. Thus, each sample is $\mathbf{x} = (\mathbf{a}, \mathbf{u})$, where $\mathbf{a}$ specifies the PDE instance and $\mathbf{u}$ is its solution.

For physics evaluation and guidance, we match the residual discretization to the benchmark solver. In particular, for Darcy flow we use a conservative flux-form discretization with harmonic face averaging, rather than applying a naive central-difference Laplacian to $a\nabla u$. Let $S = 32$ or 128 and $h = 1/(S-1)$. For interior grid nodes $(i,j) \in \mathcal{I}_{\text{int}}$, we define

$$\begin{aligned} r_{\text{Darcy}}(\mathbf{x})_{i,j} := -\frac{1}{h^2}\Big[& \mathbf{a}_{i+\frac{1}{2},j}(\mathbf{u}_{i+1,j} - \mathbf{u}_{i,j}) - \mathbf{a}_{i-\frac{1}{2},j}(\mathbf{u}_{i,j} - \mathbf{u}_{i-1,j}) \\ & + \mathbf{a}_{i,j+\frac{1}{2}}(\mathbf{u}_{i,j+1} - \mathbf{u}_{i,j}) - \mathbf{a}_{i,j-\frac{1}{2}}(\mathbf{u}_{i,j} - \mathbf{u}_{i,j-1})\Big] - q_{i,j}, \end{aligned} \tag{12}$$

where the face coefficients are computed by harmonic averaging:

$$\mathbf{a}_{i+\frac{1}{2},j} = \frac{2\mathbf{a}_{i,j}\mathbf{a}_{i+1,j}}{\mathbf{a}_{i,j} + \mathbf{a}_{i+1,j}}, \qquad \mathbf{a}_{i,j+\frac{1}{2}} = \frac{2\mathbf{a}_{i,j}\mathbf{a}_{i,j+1}}{\mathbf{a}_{i,j} + \mathbf{a}_{i,j+1}}, \tag{13}$$

with analogous definitions for $\mathbf{a}_{i-\frac{1}{2},j}$ and $\mathbf{a}_{i,j-\frac{1}{2}}$. The homogeneous Dirichlet boundary values are imposed explicitly, and the PDE error is evaluated only on interior nodes $\mathcal{I}_{\text{int}}$ to avoid boundary-discretization artifacts.

A.2 Poisson Equation

We consider the static Poisson equation on $\Omega_x = (0,1)^2$ with homogeneous Dirichlet boundary:

$$\begin{cases} \nabla^2 u(\mathbf{c}) = a(\mathbf{c}), & \mathbf{c} \in \Omega_x, \\ u(\mathbf{c}) = 0, & \mathbf{c} \in \partial\Omega_x. \end{cases} \tag{14}$$

Each dataset sample is generated by first drawing a random forcing field and then solving the PDE. Following Huang et al. [16], we set $a = \mu$ where μ is sampled from the Gaussian random field. We solve for u using a second-order finite-difference scheme, enforcing the homogeneous Dirichlet boundary via the standard mollifier $\sin(\pi c_1)\sin(\pi c_2)$ in the data-generation pipeline. We store the discretized fields on a 32×32 grid and denote each sample by $\mathbf{x} = (\mathbf{a}, \mathbf{u})$.

On $\mathcal{D}_h \subset \Omega_x$, we evaluate the benchmark-consistent discrete PDE error map $\mathcal{G}_h$ using the second-order Laplacian, yielding the interior PDE error

$$r_{\text{Pois}}(\mathbf{x})_i := (\Delta_h \mathbf{u})_i - \mathbf{a}_i, \qquad i \in \mathcal{I}_{\text{int}}, \tag{15}$$

where Δ_h is the standard second-order discrete Laplacian on the uniform grid.

A.3 Inhomogeneous Helmholtz Equation

We consider the static inhomogeneous Helmholtz equation on $\Omega_x = (0,1)^2$ with homogeneous Dirichlet boundary:

$$\begin{cases} \nabla^2 u(\mathbf{c}) + k^2 u(\mathbf{c}) = a(\mathbf{c}), & \mathbf{c} \in \Omega_x, \\ u(\mathbf{c}) = 0, & \mathbf{c} \in \partial\Omega_x, \end{cases} \tag{16}$$

where we set $k = 1$ (note that this reduces to the Poisson equation when $k = 0$).

This benchmark follows the same generation pipeline as Poisson and we store each sample $\mathbf{x} = (\mathbf{a}, \mathbf{u})$ on a 128×128 grid.

Accordingly, the discrete PDE error differs from Poisson only by the additional reaction term:

$$r_{\text{Helm}}(\mathbf{x})_i := (\Delta_h \mathbf{u})_i + k^2 \mathbf{u}_i - \mathbf{a}_i, \qquad i \in \mathcal{I}_{\text{int}}, \tag{17}$$

with the same Δ_h as above.

A.4 Non-bounded Navier–Stokes Equation

We study the 2D incompressible Navier–Stokes benchmark on $\Omega_x = (0,1)^2$ with periodic (non-bounded) boundary conditions in the vorticity formulation:

$$\begin{cases} \partial_\tau \omega(\mathbf{c},\tau) + \mathbf{v}(\mathbf{c},\tau)\cdot\nabla\omega(\mathbf{c},\tau) = \nu\,\Delta\omega(\mathbf{c},\tau) + q(\mathbf{c}), & (\mathbf{c},\tau) \in \Omega_x \times (0,T], \\ \nabla\cdot\mathbf{v}(\mathbf{c},\tau) = 0, & (\mathbf{c},\tau) \in \Omega_x \times (0,T], \end{cases} \tag{18}$$

where $\nu = 10^{-3}$, and $\omega = \nabla \times \mathbf{v}$ is the scalar vorticity. The forcing follows the fixed pattern

$$q(\mathbf{c}) = \frac{1}{10}\Big(\sin\big(2\pi(c_1 + c_2)\big) + \cos\big(2\pi(c_1 + c_2)\big)\Big). \tag{19}$$

Following Huang et al. [16], the initial condition is sampled from a Gaussian random field

$$\omega_0 \sim \mathcal{N}\big(0,\ 7^{1.5}(-\Delta + 49\mathbf{I})^{-2.5}\big). \tag{20}$$

Given ω_0 and the fixed forcing q, we solve the dynamics using a pseudo-spectral method in the stream-function formulation: we advance the vorticity equation in the Fourier domain and evaluate nonlinear terms in physical space via inverse FFTs. We simulate for 1 second using 10 time steps. In the dataset convention of DiffusionPDE, we take the terminal snapshot index as $T = 10$. We store (ω_0, ω_T) on a 64×64 spatial grid and denote the resulting discrete sample by $\mathbf{x} = (\boldsymbol{\omega}_0, \boldsymbol{\omega}_T)$, i.e., $\mathbf{a} = \boldsymbol{\omega}_0$ and $\mathbf{u} = \boldsymbol{\omega}_T$, yielding resolution $64 \times 64 \times 2$.

As in Huang et al. [16], since T is large in the benchmark convention, we do not evaluate the full time-dependent PDE error at the terminal state. Instead, we adopt a simplified terminal-time constraint evaluated directly on the predicted vorticity snapshot as terminal vorticity-gradient surrogate and define

$$r_{\mathrm{NS}}(\mathbf{x})_{i,j} := (\partial_{x,h}\boldsymbol{\omega}_T)_{i,j} + (\partial_{y,h}\boldsymbol{\omega}_T)_{i,j}, \qquad (i,j) \in \mathcal{I}_{\mathrm{all}}, \tag{21}$$

where $\partial_{x,h}$ and $\partial_{y,h}$ are central differences.

A.5 Burgers' Equation

We consider the 1D viscous Burgers' equation on $\Omega_x = (0,1)$ with periodic boundary conditions:

$$\begin{cases} \partial_\tau u(c,\tau) + \partial_c\left(\dfrac{u(c,\tau)^2}{2}\right) - \nu\,\partial_{cc}u(c,\tau) = 0, & (c,\tau) \in \Omega_x \times (0,T], \\ u(c,0) = u_0(c), & c \in \Omega_x, \end{cases} \tag{22}$$

where $\nu = 0.01$, we take the physical simulation horizon $T = 1$, and enforce periodicity in space (e.g., $u(0,\tau) = u(1,\tau)$ for all $\tau \in (0,T]$).

Following Huang et al. [16] and the FNO benchmark procedure, the initial condition is sampled as a Gaussian random field

$$u_0 \sim \mathcal{N}\big(0,\ 625(-\Delta + 25\mathbf{I})^{-2}\big). \tag{23}$$

Given u_0, we solve the PDE using a spectral method and simulate over $t \in [0,1]$ with 128 uniformly spaced time snapshots (including $t = 0$), producing a space–time solution field $u_{0:T}$. We store $\mathbf{u} = u_{0:T}$ on a 128×128 tensor grid over (c,τ).

Since the time dimension is densely discretized, we compute the pointwise PDE error by finite differences:

$$r_{\mathrm{Burg}}(\mathbf{x})_{i,j} := (\partial_{\tau,h}\mathbf{u})_{i,j} + \big(\partial_{c,h}(\tfrac{1}{2}\mathbf{u}^2)\big)_{i,j} - \nu\,(\partial_{cc,h}\mathbf{u})_{i,j}, \qquad (i,j) \in \mathcal{I}_{\mathrm{all}}, \tag{24}$$

where $\partial_{\tau,h}$, $\partial_{c,h}$ and $\partial_{cc,h}$ are standard discrete operators with periodic wrapping in space.

B Proof of Theorem 4.1

Proof of Theorem 4.1. We first justify the penalized objective used in the analysis. The constrained formulation (7) is not optimized directly in our implementation. Instead, we adopt a penalty relaxation and treat $\lambda_{\mathrm{phys}} > 0$ as a trade-off hyperparameter, which augments the IKL matching term by a soft penalty on the constraint violation. Concretely, consider the Lagrangian-type surrogate

$$\mathcal{D}_{\mathrm{IKL}}^{[0,T]}(q_\theta \| p) + \lambda_{\mathrm{phys}}\Big(\mathbb{E}_{\mathbf{x}_0 \sim q_\theta}\big[\mathcal{R}(\mathbf{x}_0)\big] - \varepsilon\Big).$$

Since $-\lambda_{\mathrm{phys}}\varepsilon$ is a constant independent of θ, it does not affect the gradient with respect to θ and can be dropped. Moreover, under our unified k-step generator parameterization, a clean student sample satisfies $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$ with $\mathbf{z} \sim p_{\mathbf{z}}$ and $k \sim \Delta_K$, which induces the implicit distribution q_θ. Therefore,

$$\mathbb{E}_{\mathbf{x}_0 \sim q_\theta}\big[\mathcal{R}(\mathbf{x}_0)\big] = \mathbb{E}_{\mathbf{z} \sim p_z,\, k \sim \Delta_K, \mathbf{x}_0 = g_\theta(\mathbf{z};k)}\Big[\mathcal{R}\big(\mathbf{x}_0\big)\Big].$$

Combining the above yields the implemented penalized objective

$$\mathcal{L}_G(\theta) = \mathcal{D}_{\mathrm{IKL}}^{[0,T]}(q_\theta \| p) + \lambda_{\mathrm{phys}}\,\mathbb{E}_{\mathbf{z} \sim p_z,\, k \sim \Delta_K, \mathbf{x}_0 = g_\theta(\mathbf{z};k)}\Big[\mathcal{R}\big(\mathbf{x}_0\big)\Big],$$

which is exactly the objective optimized by the guided generator update.

Throughout, we assume mild regularity so that differentiation can be interchanged with the time integral and expectations, and the sampling procedure that generates $\mathbf{x}_t \sim p_{0t}(\cdot \mid \mathbf{x}_0)$ admits a

pathwise derivative with respect to $\mathbf{x}_0$ (and thus with respect to θ through $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$). We also work under the ideal score limit in the theorem, i.e., $s_\phi(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log q_{\theta,t}(\mathbf{x}_t)$ and $s_\nu(\mathbf{x}_t, t) = \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ for a.e. $t \in [0, T]$ and $\mathbf{x}_t \sim q_{\theta,t}$.

Fix $t \in [0, T]$. The time-t KL term is

$$\mathcal{D}_{\mathrm{KL}}(q_{\theta,t} \| p_t) = \mathbb{E}_{\mathbf{x}_t \sim q_{\theta,t}}\big[\log q_{\theta,t}(\mathbf{x}_t) - \log p_t(\mathbf{x}_t)\big].$$

By the sampling procedure in (9), $\mathbf{x}_t$ is obtained by drawing $k \sim \Delta_K$, $\mathbf{z} \sim p_{\mathbf{z}}$, setting $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$, and sampling $\mathbf{x}_t \sim p_{0t}(\cdot \mid \mathbf{x}_0)$. Under the above regularity, differentiating the KL term along this sampling path yields

$$\nabla_\theta \mathcal{D}_{\mathrm{KL}}(q_{\theta,t} \| p_t) = \mathbb{E}\Big[\nabla_\theta \log q_{\theta,t}(\mathbf{x}_t) + \big(\nabla_{\mathbf{x}_t} \log q_{\theta,t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\big)^\top \nabla_\theta \mathbf{x}_t\Big], \tag{25}$$

where the expectation is over $k \sim \Delta_K$, $\mathbf{z} \sim p_{\mathbf{z}}$, $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$, and $\mathbf{x}_t \sim p_{0t}(\cdot \mid \mathbf{x}_0)$. The first term vanishes by the score identity $\mathbb{E}_{\mathbf{x}_t \sim q_{\theta,t}}[\nabla_\theta \log q_{\theta,t}(\mathbf{x}_t)] = 0$. Therefore,

$$\nabla_\theta \mathcal{D}_{\mathrm{KL}}(q_{\theta,t} \| p_t) = \mathbb{E}\Big[\big(\nabla_{\mathbf{x}_t} \log q_{\theta,t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\big)^\top \nabla_\theta \mathbf{x}_t\Big]. \tag{26}$$

Recalling $\mathcal{D}_{\mathrm{IKL}}^{[0,T]}(q_\theta \| p) = \int_0^T w(t)\, \mathcal{D}_{\mathrm{KL}}(q_{\theta,t} \| p_t)\, \mathrm{d}t$ and interchanging ∇_θ with the integral, we obtain

$$\nabla_\theta \mathcal{D}_{\mathrm{IKL}}^{[0,T]}(q_\theta \| p) = \int_0^T w(t)\, \mathbb{E}\Big[\big(\nabla_{\mathbf{x}_t} \log q_{\theta,t}(\mathbf{x}_t) - \nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)\big)^\top \nabla_\theta \mathbf{x}_t\Big] \mathrm{d}t. \tag{27}$$

Under the ideal score limit, $\nabla_{\mathbf{x}_t} \log q_{\theta,t}(\mathbf{x}_t) = s_\phi(\mathbf{x}_t, t)$ and $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t) = s_\nu(\mathbf{x}_t, t)$, so (27) becomes

$$\nabla_\theta \mathcal{D}_{\mathrm{IKL}}^{[0,T]}(q_\theta \| p) = \int_0^T w(t)\, \mathbb{E}\Big[\big(s_\phi(\mathbf{x}_t, t) - s_\nu(\mathbf{x}_t, t)\big)^\top \nabla_\theta \mathbf{x}_t\Big] \mathrm{d}t. \tag{28}$$

In the implemented update, the score term is treated as a constant weight with respect to θ, i.e., gradients are stopped through the score evaluations. Equivalently, we may insert the stop-gradient operator $\mathrm{SG}\{\cdot\}$ on the score difference inside the expectation in (28) without changing the resulting θ-gradient, since the only θ-dependence being differentiated is carried by the pathwise term $\nabla_\theta x_t$. Therefore,

$$\nabla_\theta \mathcal{D}_{\mathrm{IKL}}^{[0,T]}(q_\theta \| p) = \int_0^T w(t)\, \mathbb{E}\Big[\mathrm{SG}\big\{s_\phi(\mathbf{x}_t, t) - s_\nu(\mathbf{x}_t, t)\big\}^\top \nabla_\theta \mathbf{x}_t\Big]\, dt, \tag{29}$$

where the expectation is over $\mathbf{z} \sim p_{\mathbf{z}}$, $k \sim \Delta_K$, $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$, and $\mathbf{x}_t \sim p_{0t}(\cdot \mid \mathbf{x}_0)$.

For the physics term, recall from (9) that the penalty enters as $\lambda_{\mathrm{phys}} \mathcal{L}_{\mathrm{phys}}(\theta)$, where

$$\mathcal{L}_{\mathrm{phys}}(\theta) = \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}},\, k \sim \Delta_K,\, \mathbf{x}_0 = g_\theta(\mathbf{z}; k)}\big[\mathcal{R}(\mathbf{x}_0)\big].$$

Under the same regularity assumptions, the chain rule yields

$$\nabla_\theta \mathcal{L}_{\mathrm{phys}}(\theta) = \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}},\, k \sim \Delta_K}\Big[\big(\nabla_{\mathbf{x}_0} \mathcal{R}(\mathbf{x}_0)\big)^\top \nabla_\theta g_\theta(\mathbf{z}; k)\Big]_{\mathbf{x}_0 = g_\theta(\mathbf{z}; k)}. \tag{30}$$

Combining (29) with (30) gives

$$\begin{aligned}\nabla_\theta \mathcal{L}_G(\theta) = \int_0^T w(t)\, \mathbb{E}_{\substack{\mathbf{z} \sim p_{\mathbf{z}},\, k \sim \Delta_K \\ \mathbf{x}_0 = g_\theta(\mathbf{z}; k),\, \mathbf{x}_t \sim p_{0t}(\cdot \mid \mathbf{x}_0)}} \Big[\mathrm{SG}\big\{s_\phi(\mathbf{x}_t, t) - s_\nu(\mathbf{x}_t, t)\big\}^\top \nabla_\theta \mathbf{x}_t\Big] \mathrm{d}t \\ + \ \lambda_{\mathrm{phys}} \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}},\, k \sim \Delta_K}\Big[\big(\nabla_{\mathbf{x}_0} \mathcal{R}(\mathbf{x}_0)\big)^\top \nabla_\theta g_\theta(\mathbf{z}; k)\Big]_{\mathbf{x}_0 = g_\theta(\mathbf{z}; k)},\end{aligned}$$

which is exactly (10). This completes the proof. □

C Phys-Instruct Algorithm

Algorithm 1 summarizes the training procedure of Phys-Instruct. The pretrained teacher score model is kept frozen, while the auxiliary score model and the student generator are updated alternately; the physics guidance is applied only to the generator update.

Algorithm 1 Phys-Instruct.

Input: pre-trained model s_ν; generator g_θ; prior p_z; auxiliary s_ϕ; regularization weight λ_{phys}; max step K.
while not converge **do**
  // Score-learning phase for auxiliary diffusion model
  Sample $\mathbf{z} \sim p_z$, $k \sim \Delta_K$, set $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$ and diffuse to $\mathbf{x}_t$ via the forward SDE (2).
  Update ϕ using SGD with loss function (3).
  // Instruction + physics phase for generator
  Sample $\mathbf{z} \sim p_z$, $k \sim \Delta_K$, set $\mathbf{x}_0 = g_\theta(\mathbf{z}; k)$ and diffuse to $\mathbf{x}_t$ via the forward SDE (2).
  Update θ using SGD with loss function (9), whose tractable gradient is (10).
end while
Output: θ, ϕ.

D Distributional Metrics

In addition to PDE Error, we report distributional discrepancy between generated samples and held-out real data using two-sample distances. These metrics quantify how closely the generated fields match the data distribution, complementing physics-consistency evaluations.

Preprocessing and Normalization. All distances are computed on the same field representation used for evaluation. For multi-field problems (e.g., coefficient–solution pairs), we apply channel-wise standardization using statistics computed from the real data: for each channel c, we transform both real and generated samples by

$$\tilde{\mathbf{x}}^{(c)} = \frac{\mathbf{x}^{(c)} - \mu^{(c)}_{\text{real}}}{\sigma^{(c)}_{\text{real}} + \epsilon}, \tag{31}$$

where $\mu^{(c)}_{\text{real}}$ and $\sigma^{(c)}_{\text{real}}$ are the mean and standard deviation over the real dataset, and ϵ is a small constant. For single-field benchmarks such as Burgers' equation, this reduces to standardization of the scalar field.

Sliced Wasserstein Distance (SWD) [53, 21]. Let P and Q denote the empirical distributions of standardized generated and real samples, respectively. SWD approximates the Wasserstein distance in high dimensions by averaging 1D Wasserstein distances over random projections:

$$\text{SWD}(P, Q) = \frac{1}{K} \sum_{k=1}^{K} W_1(\langle \mathbf{v}_k, \mathbf{x} \rangle_{\mathbf{x} \sim P},\ \langle \mathbf{v}_k, \mathbf{x} \rangle_{\mathbf{x} \sim Q}), \tag{32}$$

where $\{\mathbf{v}_k\}_{k=1}^K$ are i.i.d. random unit vectors and $W_1(\cdot, \cdot)$ is the 1D Wasserstein distance computed via sorting. We use a fixed random seed and $K = 512$ random projections, which provides a stable estimate of SWD at a moderate computational cost for high-dimensional fields.

Maximum Mean Discrepancy (MMD) [48]. We also report MMD with a multi-scale RBF kernel as a complementary two-sample statistic:

$$\text{MMD}^2(P, Q) = \mathbb{E}_{\mathbf{x}, \mathbf{x}' \sim P}[k(\mathbf{x}, \mathbf{x}')] + \mathbb{E}_{\mathbf{y}, \mathbf{y}' \sim Q}[k(\mathbf{y}, \mathbf{y}')] - 2\,\mathbb{E}_{\mathbf{x} \sim P, \mathbf{y} \sim Q}[k(\mathbf{x}, \mathbf{y})], \tag{33}$$

with $k(\mathbf{x}, \mathbf{y}) = \exp\big(-\|\mathbf{x} - \mathbf{y}\|_2^2 / (2\sigma^2)\big)$. Bandwidths are selected from the real data using the median pairwise-distance heuristic: we compute $\hat{\sigma}$ as the median of pairwise distances between a random subset of real samples, and use the multi-scale set $\sigma \in \{\hat{\sigma}/4,\ \hat{\sigma}/2,\ \hat{\sigma},\ 2\hat{\sigma},\ 4\hat{\sigma}\}$. We report $\sqrt{\text{MMD}^2}$ for readability.

D.1 Additional Distributional Coverage and Sample Diversity on Burgers' Equation

To further examine what is preserved by physics-guided distillation, we complement the SWD and MMD results in Figure 3 with distributional coverage and sample-diversity measurements on Burgers' equation. We extend the student sampling budget to $k \in \{1, 2, 4, 8\}$ and compare against the 200-step EDM teacher. Here kNN Recall measures support coverage, and Diversity/T200 measures sample-to-sample variation relative to the 200-step teacher, whose value is normalized to 1.

Table 4 shows that one-step distillation reduces both support coverage and sample variation, despite substantially improving PDE Error. Most of the coverage gap is recovered by 4 steps: Recall increases from 0.674 to 0.930, compared to 0.980 for EDM-200, while Diversity/T200 increases from 0.658 to 0.890. At the same time, SWD approaches the teacher value (0.049 versus 0.043), and PDE Error remains substantially lower (0.055 versus 0.860). Increasing the budget to 8 steps further improves PDE Error and SWD, but yields only a small increase in relative diversity and a slight decrease in Recall. On this benchmark, most of the coverage improvement is obtained by $k = 4$, while the additional gains at $k = 8$ are smaller.

Table 4: Additional distributional coverage and sample diversity on Burgers' equation.

Method	Step	Inference Latency (s)↓	PDE Error↓	SWD↓	Recall↑	Diversity/T200
Phys-Instruct	1	0.012	0.171	0.071	0.674	0.658
	2	0.036	0.118	0.060	0.830	0.896
	4	0.084	0.055	0.049	0.930	0.890
	8	0.225	0.046	0.046	0.919	0.905
EDM	200	4.919	0.860	0.043	0.980	1.000

E Normalized Compute and Update-Level Breakdown

To complement the wall-clock discussion, we report normalized training compute. Since teacher pretraining and Phys-Instruct distillation were conducted on different GPU hardware, raw runtime is not directly comparable across stages. We therefore use FLOPs-based measurements when such profiling is available.

Table 5 summarizes the total training compute. For Darcy Flow, Poisson, and non-bounded Navier–Stokes, where both teacher and distillation were profiled in our setup, Phys-Instruct requires about 74% of the teacher-training compute. For Burgers' and Helmholtz, the teacher checkpoints were inherited from prior work rather than retrained in-house. We therefore report only the measured distillation compute for these two benchmarks and omit teacher-side totals and D/T ratios.

Table 6 further decomposes the per-update distillation cost into the auxiliary score-model update (s_ϕ), the generator update (g_θ), and the PDE-guidance portion within g_θ. Across all benchmarks, the overall cost is dominated by the generator-side update, while the PDE residual contributes a stable overhead of about one-third of the generator-update cost. This indicates that the added physics guidance remains bounded and is not the primary computational bottleneck.

F Additional Phys-Instruct Generation Results

F.1 Sample-Level Variability

To complement the mean PDE Error reported in Table 1, we additionally compute the sample standard deviation of the per-sample PDE Error for Phys-Instruct in Table 7. The statistics are evaluated using 1000 randomly generated samples for each benchmark and sampling budget. Across all five benchmarks, increasing the sampling budget from 1 to 4 steps reduces the mean PDE Error. The sample standard deviation also generally decreases with more sampling steps, indicating that the improvement is not only reflected in the average error but also in more concentrated sample-level PDE errors.

F.2 Resolution Scalability on Poisson

To further examine scalability with respect to spatial resolution, we additionally evaluate one-step Phys-Instruct on the Poisson benchmark at resolutions $128\times128\times2$, $256\times256\times2$, and $512\times512\times2$, complementing the $32 \times 32 \times 2$ result reported in the main submission. All experiments use the same data-generation and evaluation pipeline.

Table 8 summarizes the PDE Error, the ratio of student PDE Error to teacher PDE Error, the per-sample inference time, and the peak GPU memory. Since absolute PDE residuals naturally increase as the grid becomes finer, the more informative quantity across resolutions is the ratio to the multi-step

Table 5: Normalized total training compute in kTFLOPs. D/T denotes the ratio between Phys-Instruct distillation cost and teacher-training cost. For Burgers' and Helmholtz, the teacher checkpoints were inherited from prior work, so comparable teacher-training compute is not reported.

Benchmark	Teacher	Distill	D/T
Darcy Flow	689.4	506.9	0.735
Poisson	689.4	506.9	0.735
$\text{NS}_{\text{unb.}}$	1609.0	1191.1	0.740
Burgers'	—	3185.3	—
Helmholtz	—	3364.9	—

Table 6: Update-level compute breakdown of Phys-Instruct distillation, reported in TFLOPs and normalized by batch size. Here, PDE denotes the portion of the generator update attributable to PDE guidance, so Total corresponds to $s_\phi + g_\theta$, while PDE/G reports the fraction of g_θ consumed by PDE guidance.

Benchmark	s_ϕ	g_θ	PDE	PDE/G	Total
Darcy	0.086	0.128	0.042	33.2%	0.213
Poisson	0.086	0.128	0.042	33.2%	0.213
$\text{NS}_{\text{unb.}}$	0.269	0.440	0.141	32.0%	0.709
Burgers'	1.328	1.990	0.663	33.3%	3.318
Helmholtz	1.328	1.991	0.663	33.3%	3.319

teacher. This ratio remains well below 1 at all tested resolutions, indicating that the one-step student consistently preserves its advantage over the teacher as problem size increases. In addition, inference remains practical on a single A100 GPU even at $512 \times 512 \times 2$, requiring 317.00 ms and 3075.57 MB per sample.

F.3 Qualitative Results of Darcy Flow

We show the generated Darcy flow samples with resolution 32×32 under 1-4 steps with Phys-Instruct in Figure 6. With sampling steps increasing, PDE error drops.

F.4 Physical Diagnostics Beyond PDE Error

PDE Error measures consistency with the benchmark-specific discrete equation, but does not alone characterize solution accuracy or other physical properties. We therefore evaluate complementary diagnostics on Poisson and Burgers' equations and an additional time-dependent Kolmogorov-flow benchmark. These experiments examine consistency under independent discretizations, independently computed numerical solutions, conservation and balance laws, and spectral structure.

Poisson: Held-Out Discretization and Independent Re-Solving. For each unconditionally generated pair $(\mathbf{a}, \mathbf{u})$, we independently solve the homogeneous Dirichlet Poisson problem using the generated forcing $\mathbf{a}$. We compare the generated solution with this numerical solution using the

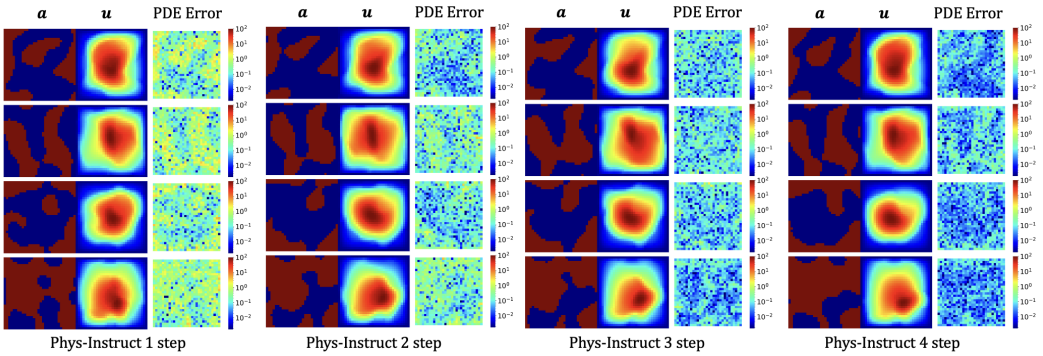


Figure 6: Phys-Instruct generated Darcy flow samples with 1-4 steps (heun sampler).The PDE error column is showing the absolute value of the PDE error field $r(\mathbf{x})_i$ for every point in the grid.

Table 7: Sample-level variability of Phys-Instruct PDE Error over 1000 generated samples. We report mean $\pm$ standard deviation.

Benchmark	Resolution	Step	PDE Error
Darcy Flow	$32 \times 32 \times 2$	1	$3.15\times10^{-1} \pm 6.81\times10^{-2}$
		2	$2.18\times10^{-1} \pm 5.27\times10^{-2}$
		3	$1.35\times10^{-1} \pm 3.83\times10^{-2}$
		4	$8.49\times10^{-2} \pm 2.13\times10^{-2}$
Poisson	$32 \times 32 \times 2$	1	$2.16\times10^{-2} \pm 5.57\times10^{-3}$
		2	$5.18\times10^{-3} \pm 4.89\times10^{-3}$
		3	$4.79\times10^{-3} \pm 3.86\times10^{-3}$
		4	$3.99\times10^{-3} \pm 3.55\times10^{-3}$
$\mathrm{NS}_{\mathrm{unb.}}$	$64 \times 64 \times 2$	1	$3.15\times10^{-2} \pm 6.74\times10^{-3}$
		2	$1.46\times10^{-2} \pm 5.19\times10^{-3}$
		3	$3.65\times10^{-3} \pm 4.78\times10^{-3}$
		4	$3.21\times10^{-3} \pm 4.30\times10^{-3}$
Burgers'	$128 \times 128 \times 1$	1	$1.71\times10^{-1} \pm 3.86\times10^{-2}$
		2	$1.18\times10^{-1} \pm 3.05\times10^{-2}$
		3	$7.17\times10^{-2} \pm 2.74\times10^{-2}$
		4	$5.54\times10^{-2} \pm 2.20\times10^{-2}$
Helmholtz	$128 \times 128 \times 2$	1	$3.75\times10^{-1} \pm 1.14\times10^{-1}$
		2	$3.00\times10^{-1} \pm 1.12\times10^{-1}$
		3	$2.10\times10^{-1} \pm 8.66\times10^{-2}$
		4	$1.46\times10^{-1} \pm 6.84\times10^{-2}$

Table 8: Resolution scalability of one-step Phys-Instruct on the Poisson benchmark. "Ratio to Teacher" denotes the PDE Error of Phys-Instruct divided by that of the multi-step teacher; lower is better. Inference time and peak memory are measured on a single A100 GPU.

Resolution	Step	PDE Error	Ratio to Teacher $\downarrow$	Time (ms)	Peak Mem (MB)
$32 \times 32 \times 2$	1	0.02	69%	16.87	357.37
$128 \times 128 \times 2$	1	0.52	41%	19.38	452.95
$256 \times 256 \times 2$	1	3.52	59%	62.59	894.08
$512 \times 512 \times 2$	1	12.33	34%	317.00	3075.57

re-solve L^2 and H^1 errors to assess solution and gradient accuracy. We also evaluate a held-out fourth-order finite-difference residual (FD4), which is not used for physics guidance during distillation.

Table 9 shows that the one-step student has a lower FD4 residual than EDM-200, but larger re-solve errors. At 4 steps, Phys-Instruct improves all three diagnostics, indicating that the gain extends beyond the residual discretization used for training to independently evaluated solution values and gradients.

Table 9: Independent physical diagnostics for unconditional Poisson generation. Lower is better for all metrics.

Method	Step	FD4 Residual	Re-solve L^2	Re-solve H^1
Phys-Instruct	1	2.76×10^{-2}	2.68×10^{-2}	3.66×10^{-2}
	4	$\mathbf{5.76\times10^{-3}}$	$\mathbf{6.74\times10^{-3}}$	$\mathbf{6.49\times10^{-3}}$
EDM	200	4.23×10^{-2}	1.25×10^{-2}	1.51×10^{-2}

Burgers': Spectral Residual, Mass Conservation, and Energy Balance. For the periodic viscous Burgers' equation, we additionally evaluate an independent spectral residual, a mass-conservation defect, and a viscous energy-balance defect. The latter two diagnostics test the identities

$$\frac{\mathrm{d}}{\mathrm{d}\tau}\int_{\Omega_x} u(c,\tau)\,\mathrm{d}c = 0, \qquad \frac{\mathrm{d}}{\mathrm{d}\tau}\frac{1}{2}\int_{\Omega_x} u(c,\tau)^2\,\mathrm{d}c + \nu\int_{\Omega_x} |\partial_c u(c,\tau)|^2\,\mathrm{d}c = 0,$$

where $\nu = 0.01$ as in Appendix A. The energy diagnostic accounts for viscous dissipation rather than assuming energy conservation.

As shown in Table 10, 4-step Phys-Instruct achieves lower spectral residual, mass defect, and energy-balance defect than EDM-200. The spectral residual does not decrease monotonically with the student budget, further illustrating that the independent diagnostics capture different properties from the training residual. We also compare the generated trajectory with an independent numerical rollout initialized from its generated initial frame. Four-step Phys-Instruct and EDM-200 have similar rollout errors (0.0300 and 0.0309, respectively), while the student improves the conservation and balance diagnostics.

Table 10: Independent physical diagnostics on Burgers' equation. The spectral residual uses an evaluator distinct from the finite-difference residual used for physics guidance. Lower is better for all metrics.

Method	Step	Spectral Residual	Mass Defect	Energy-Balance Defect
Phys-Instruct	1	$\mathbf{1.136}$	3.54×10^{-2}	1.968×10^{-1}
	4	1.194	$\mathbf{1.19\times10^{-2}}$	$\mathbf{1.016\times10^{-1}}$
EDM	200	1.630	1.34×10^{-2}	2.969×10^{-1}

Kolmogorov Flow: Temporal and Spectral Diagnostics. To examine a two-dimensional time-dependent flow, we use the released $\mathrm{Re} = 1000$ Kolmogorov-flow dataset of Shu et al. [42][2]. The dataset contains periodic vorticity fields on $(0, 2\pi)^2$, stored on a 256×256 grid with physical-time spacing $\Delta\tau = 1/32$. The governing equation includes advection, viscosity, sinusoidal forcing, and linear damping:

$$\partial_\tau \omega + \mathbf{v} \cdot \nabla\omega = \mathrm{Re}^{-1}\Delta\omega - 4\cos(4c_2) - 0.1\omega, \qquad \nabla \cdot \mathbf{v} = 0. \tag{34}$$

Here, $\mathbf{v}$ denotes the velocity recovered from the vorticity using the periodic stream-function relation. We convert the trajectories into overlapping 3-frame samples $\mathbf{x} = (\boldsymbol{\omega}_{\tau-\Delta\tau}, \boldsymbol{\omega}_\tau, \boldsymbol{\omega}_{\tau+\Delta\tau})$, and train an EDM teacher on this representation before distilling an 8-step Phys-Instruct student. The original Navier–Stokes benchmark uses initial and terminal snapshots; this additional experiment instead evaluates the time-dependent equation locally through three consecutive frames.

For each generated three-frame sample, we evaluate the residual at the middle frame:

$$\begin{aligned} r_{\mathrm{Kolm}}(\mathbf{x})_{i,j} := \; & \left[\frac{\boldsymbol{\omega}_{\tau+\Delta\tau} - \boldsymbol{\omega}_{\tau-\Delta\tau}}{2\Delta\tau} + \mathbf{v}_\tau \cdot \nabla_h \boldsymbol{\omega}_\tau - \mathrm{Re}^{-1}\Delta_h \boldsymbol{\omega}_\tau\right]_{i,j} \\ & + 4\cos(4c_{2,j}) + 0.1(\boldsymbol{\omega}_\tau)_{i,j}. \end{aligned} \tag{35}$$

Here, ∇_h and Δ_h are Fourier pseudo-spectral operators, and $c_{2,j}$ denotes the second spatial coordinate. Following (6), we compute $\mathcal{R}(\mathbf{x})$ using this residual over all 256^2 periodic grid points and report PDE Error as the sample average of $\sqrt{\mathcal{R}(\mathbf{x})}$. For an independent check, Spectral PDE Error uses the same aggregation with a residual recomputed using two-thirds dealiasing and standard zero-mode treatment. We also report radial-spectrum mismatch relative to EDM-200. Since EDM-200 defines this spectral reference, its mismatch is zero by construction; this metric measures agreement with the teacher's spectrum rather than with the held-out data spectrum. Table 11 shows that at the same 8-step budget, Phys-Instruct reduces PDE Error from 150.2 to 18.86, Spectral PDE Error from 106.13 to 17.09, and spectrum mismatch from 1.880 to 0.291.

Table 11: Unconditional-generation results on $\mathrm{Re} = 1000$ Kolmogorov flow using 3 consecutive 256×256 vorticity frames. Spectrum mismatch is measured relative to EDM-200. Lower is better for all metrics.

Method	Step	PDE Error	Spectral PDE Error	Spectrum Mismatch
EDM	8	150.2	106.13	1.880
Phys-Instruct	8	$\mathbf{18.86}$	$\mathbf{17.09}$	0.291
EDM	200	31.46	26.08	0 (reference)

[2] `https://github.com/BaratiLab/Diffusion-based-Fluid-Super-resolution`.

Table 12: Sensitivity of training stability to λ_{phys}. All runs use the same setup as **Default** except for different λ_{phys}. We label runs as *Stable* (training completes normally), *Weak* (training completes but yields limited PDE error improvement), *Unstable* (fails early before the PDE error meaningfully decreases), and *Diverged* (mode collapsed, distillation failed).

λ_{phys}	Status
$1\text{e}{-4}$	Unstable (early failure; PDE error remains high)
$5\text{e}{-4}$	Stable, but weak improvement
$1\text{e}{-3}$	Stable
$5\text{e}{-3}$	Stable
$1\text{e}{-2}$	Stable
$5\text{e}{-2}$	Diverged

G Extra Ablation Studies

This appendix provides additional ablations that complement the main-text analysis of physics guidance in one-step Phys-Instruct. Unless otherwise stated, all runs follow the **Default** configuration on Darcy flow, and we report the PDE Error evaluated on one-step samples. Moreover, we also include ablation study of step operator as described in 4.1.

G.1 Sensitivity to the Physics Guidance Weight λ_{phys}

Figure 7 (a) plots the training trajectories of the PDE Error under different physics guidance weights λ_{phys}. Across a broad and practically relevant range, physics guidance consistently accelerates the decay of the PDE error and leads to a lower final plateau. In particular, *moderate* guidance weights yield the most reliable improvements, showing both rapid early reduction and sustained progress throughout training. Figure 7 (b) further summarizes the final PDE error versus λ_{phys}, showing that a moderate range around $5\text{e}{-3}$ achieves the lowest PDE errors.

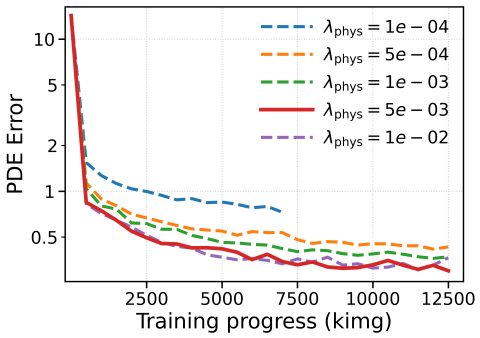
When λ_{phys} is too small, the physics signal becomes ineffective: training either fails to make meaningful progress or plateaus at a noticeably higher PDE error. Conversely, overly large λ_{phys} can destabilize training and lead to divergence, as summarized in Table 12. These results suggest a *stability–PDE error trade-off*: insufficient weighting under-enforces the PDE constraint, while excessive weighting can dominate the distillation objective and break optimization.

Overall, the default choice of λ_{phys} lies in a stable regime that achieves strong PDE error reduction without sacrificing training stability, and the method remains robust to moderate variations around this value.

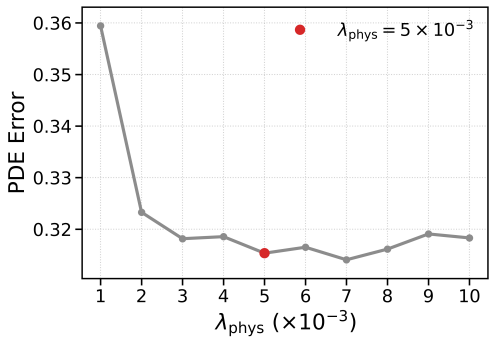
Practical Tuning of λ_{phys}. In practice, selecting an effective λ_{phys} is straightforward. The tuning process is initialized with $\lambda_{\text{phys}} = 10^{-3}$, followed by a short early-training stage that typically requires less than 15 minutes and only a limited number of iterations. The magnitudes of the distribution-matching loss and the physics-residual loss are then compared, with the goal of keeping the physics term within the same order of magnitude as the distribution-matching term so that neither component dominates optimization. If the physics term is substantially smaller, λ_{phys} is increased by a factor of 10; if it is substantially larger, λ_{phys} is decreased by the same factor. Once the appropriate order of magnitude is identified, only a minor local adjustment is typically needed. Across benchmarks, this empirical procedure is found to be simple and effective, while avoiding an expensive grid search.

G.2 Sensitivity to the Initial Noise Level σ_{init}

Table 13 studies the sensitivity of one-step Phys-Instruct to the initial noise level σ_{init}, holding all other settings fixed. We find that performance is not highly sensitive to σ_{init} within the tested range: the final PDE Error varies only mildly, indicating that the default σ_{init} is not a fragile tuning choice. This robustness is desirable in practice, as it suggests that one-step distillation with physics guidance does not require aggressive calibration of the initial noise level to achieve good physical consistency.



(a) Training dynamics.



(b) Final PDE error vs. λ_{phys}.

Figure 7: Sensitivity of PDE error to the physics guidance weight λ_{phys} on Darcy flow. (a) We plot PDE error (log scale) versus training progress for one-step Phys-Instruct with different λ_{phys} (all other settings match **Default**). Moderate λ_{phys} yields consistently lower PDE errors, while too small λ_{phys} leads to early instability and a higher error plateau. Diverged runs with overly large λ_{phys} are omitted. (b) Summary of the final PDE error across λ_{phys} with finer grids between $1e{-}3$ and $1e{-}2$. The red dot denotes the PDE error of default λ_{phys}.

Table 13: Sensitivity check of the one-step noise level σ_{init}. All settings use the default configuration except for σ_{init}.

σ_{init}	PDE Error ($\times 10^{-1}$)$\downarrow$
1.00	3.21
1.50 (default)	3.15
3.00	3.18

G.3 Ablation on Step Operator

Selecting a step operator is equivalent to specifying the few-step generator's sampling scheme. We use Heun as the default, and additionally report results with an Euler sampler on Darcy flow. As shown in Table 14, switching between Heun and Euler changes Phys-Instruct's performance only marginally, since both employ a fixed, small-step trajectory.

H Conditional Downstream Tasks with ControlNet

The unconditional Phys-Instruct models learn a prior over PDE fields without paired supervision. In many downstream applications, however, we are given paired data

$$\{(\mathbf{y}_i, \mathbf{x}_i^\star)\}_{i=1}^N, \tag{36}$$

where $\mathbf{y}_i$ encodes problem-specific information, such as coefficient fields, boundary conditions, or observation masks, and $\mathbf{x}_i^\star$ is the corresponding ground-truth PDE field (solution and, when applicable, coefficients). Conditional Phys-Instruct leverages this supervision while still using the PDE error as physics guidance.

ControlNet-Style Conditional Generator. We build a conditional generator on top of the unconditional Phys-Instruct backbone,

$$\hat{\mathbf{x}} = g_\theta(\mathbf{z}, \mathbf{y}), \qquad \mathbf{z} \sim p_z, \tag{37}$$

where $\mathbf{y}$ provides task-specific conditioning. For conditional downstream tasks, we instantiate g_θ using the one-step Phys-Instruct generator as the backbone, so the explicit sampling step index k is omitted. Our backbones are EDM following the SongUNet architecture [19, 46], consisting of an encoder–decoder U-Net with multi-resolution residual blocks and skip connections.

Following Zhang et al. [62], we attach a *control branch* that processes $\mathbf{y}$ and injects its features into the backbone through zero-initialized residual connections. Let $\mathbf{e}$ denote the shared timestep (noise-level) embedding produced by the mapping network. We first map the condition $\mathbf{y}$ to the channel

Table 14: Unconditional generation results on Darcy flow at resolution $32\times32\times2$ with two different sampling methods. We report PDE Error↓ and the number of sampling steps per sample (Step). The lowest PDE Error within each sampler column is **bolded**.

Method	Step	PDE Error (↓)	
		Euler	Heun
Phys-Instruct	1	3.15×10^{-1}	3.15×10^{-1}
	2	2.31×10^{-1}	2.18×10^{-1}
	3	1.20×10^{-1}	1.35×10^{-1}
	4	8.94×10^{-2}	$\mathbf{8.49\times10^{-2}}$

space of the backbone input $\mathbf{z}$ using a convolutional encoder ψ, and form a condition-augmented input for the control branch:

$$\mathbf{z}^c = \mathbf{z} + \mathrm{ZeroConv}_{\mathrm{in}}\big(\psi(\mathbf{y})\big), \tag{38}$$

where $\mathrm{ZeroConv}_{\mathrm{in}}$ is a zero-initialized 1×1 convolution, hence $\mathbf{z}^c \approx \mathbf{z}$ at initialization. The control encoder mirrors the SongUNet encoder (same resolution hierarchy and block layout) and processes $(\mathbf{z}^c, \mathbf{e})$ to produce multi-resolution hints $\{\mathbf{h}_\ell(\mathbf{z}^c, \mathbf{e})\}_\ell$ aligned with the backbone decoder resolutions. Let $\mathbf{f}_\ell$ denote the backbone feature map immediately before a decoder UNet block at resolution level ℓ. We inject the conditioning by projecting the corresponding hint with a zero-initialized 1×1 convolution Proj_ℓ and adding it to the backbone feature:

$$\mathbf{f}'_\ell = \mathbf{f}_\ell + \mathrm{Proj}_\ell\big(\mathbf{h}_\ell(\mathbf{z}^c, \mathbf{e})\big), \qquad \mathbf{f}_{\ell,\mathrm{out}} = \mathrm{Block}_\ell(\mathbf{f}'_\ell, \mathbf{e}). \tag{39}$$

Since both $\mathrm{ZeroConv}_{\mathrm{in}}$ and Proj_ℓ are zero-initialized, the injected residuals are initially inactive, so the conditional generator starts from the pretrained unconditional behavior and gradually learns to incorporate $\mathbf{y}$ through the residual pathways.

Data Term and Physics Regularization. For each pair $(\mathbf{y}, \mathbf{x}^\star)$ and latent noise $\mathbf{z}$, the conditional generator produces a prediction

$$\hat{\mathbf{x}} = g_\theta(\mathbf{z}, \mathbf{y}).$$

Depending on the task, $\mathbf{x}^\star$ may contain full fields or only partially observed components. We write $\mathcal{O}_{\mathbf{y}}(\cdot)$ for a generic observation mask that extracts the components of $\mathbf{x}$ that are supervised under condition $\mathbf{y}$.

The data-fidelity term is then

$$\mathcal{L}_{\mathrm{data}}(\theta) = \mathbb{E}_{(\mathbf{y},\mathbf{x}^\star),\,\mathbf{z}}\Big[\big\|\mathcal{O}_{\mathbf{y}}(\hat{\mathbf{x}}) - \mathcal{O}_{\mathbf{y}}(\mathbf{x}^\star)\big\|_2^2\Big]. \tag{40}$$

In addition, we evaluate the PDE error on the full prediction $\hat{\mathbf{x}}$ using the same $\mathcal{G}_h$ as in Sec. 4.2, leading to the same physics error defined in Eq. (8).

Conditional Phys-Instruct Objective. The Conditional Phys-Instruct generator is trained with a supervised objective regularized by the PDE error:

$$\mathcal{L}_G^{\mathrm{cond}}(\theta) = \mathcal{L}_{\mathrm{data}}(\theta) + \lambda_{\mathrm{phys}}\,\mathcal{L}_{\mathrm{phys}}(\theta), \tag{41}$$

where $\lambda_{\mathrm{phys}} > 0$ controls the strength of physics guidance.

This formulation allows a pretrained unconditional Phys-Instruct model to be easily and flexibly adapted to handling different downstream conditional tasks by changing the definition of $\mathbf{y}$ and the observation operator $\mathcal{O}_{\mathbf{y}}$, while the $\mathcal{L}_{\mathrm{phys}}$ term encourages physically consistent predictions.

Diffusion Baselines and Metrics. We compare against DiffusionPDE [16], CoCoGen [18], and PIDM [4]. All baselines solve conditional tasks via diffusion posterior sampling (DPS), using observation and/or PDE guidance at inference time. DiffusionPDE follows the original design with DPS guidance, CoCoGen appends an additional ten-step PDE-guided refinement after sampling, and PIDM uses the same DPS-style inference as in DiffusionPDE but replaces the backbone with an unconditional PIDM model. We report prediction error (relative L_2 for continuous fields and absolute error for discrete fields) together with PDE Error. Since DPS-based solvers typically require long trajectories to converge, we run baseline inference with Step$= 2000$ as adopted in Huang et al. [16].

Table 15: Comparison of Phys-Instruct with deterministic baselines on forward and inverse tasks for Darcy Flow and Helmholtz at resolution 128×128.

	Step	Darcy Flow		Helmholtz	
		Forward	Inverse	Forward	Inverse
Phys-Instruct (ours)	1	**0.50%**	**0.35%**	**1.86%**	**4.58%**
Transolver	–	2.5%	7.5%	2.3%	12.0%
FNO	–	5.3%	5.6%	11.1%	5.0%
PINO	–	4.0%	2.1%	4.9%	4.9%
DeepONet	–	12.3%	8.4%	17.8%	28.1%
PINNs	–	15.4%	10.1%	18.1%	29.2%

Table 16: Forward and inverse results on Darcy flow and Poisson equation at resolution 32×32. We report prediction error and PDE error (↓). For forward problems, prediction error is relative error for both Darcy flow and Poisson. For inverse problems, prediction error is absolute error for Darcy flow and relative error for Poisson. Step denotes the number of sampling steps per sample.

Method	Step	Darcy Flow				Poisson			
		Forward		Inverse		Forward		Inverse	
		Rel. Err.	PDE Err.	Abs. Err.	PDE Err.	Rel. Err.	PDE Err.	Rel. Err.	PDE Err.
Phys-Instruct	1	**0.29%**	**0.28**	**0.49%**	**0.22**	**2.49%**	**0.10**	**8.64%**	**0.015**
DiffusionPDE	2000	4.61%	0.53	7.71%	1.68	7.48%	0.22	13.34%	0.091
CoCoGen	2000	4.39%	0.45	7.44%	1.59	7.25%	0.20	13.17%	0.071
PIDM	2000	7.34%	0.34	9.83%	1.76	5.28%	0.12	17.48%	0.077

Tasks. We mainly consider three kinds of down stream tasks: (1) Forward problem: inference solution $\mathbf{u}$ from known coefficients $\mathbf{a}$; (2) Inverse problem: predict coefficient $\mathbf{a}$ given observed $\mathbf{u}$; (3) Reconstruction problem: predict the whole $\mathbf{x}^\star$ given its partial observation.

I Extra Downstream Task Results

I.1 Comparison with Deterministic Surrogate Models

We compare Phys-Instruct with deterministic surrogate models on the forward and inverse problem for Darcy flow and Helmholtz on the resolution 128×128. The deterministic baselines that we compare are Transolver [52], FNO [23], PINO [24], DeepONet [29], and PINNs [39]. For FNO, PINO, DeepONet, and PINNs, we report the quantitative values from the corresponding table in Huang et al. [16]. The results are summarized in Table 15. While the deterministic baselines are powerful in these tasks, our method still demonstrates superior performance in all 4 cases, highlighting the strong modeling capabilities of Phys-Instruct.

I.2 Additional Forward and Inverse Problem

We additionally conducted forward problem on Darcy flow and Poisson equation at resolution 32×32, using the one step Phys-Instruct in Table 1 as the unconditional backbone. We show the results in Table 16. One step Phys-Instruct achieves lowest relative/absolute error on the predicted solution, as well as the lowest PDE error.

I.3 Reconstruction Problem

We follow Huang et al. [16] and consider a Burgers' equation reconstruction setting under sparse observations. Specifically, we randomly sample 5 out of 128 spatial locations as sensors and provide their measurements over time. Given these partial observations, the goal is to recover the full 128×128 spatiotemporal field. Results are summarized in Table 17. We observe an accuracy–latency trade-off: one-step Phys-Instruct improves over baselines using 200 sampling steps in both prediction error and PDE error, while still trailing diffusion baselines with 2000 steps of roll-outs. Nevertheless, as shown in Figure 8, the one-step reconstructions preserves the dominant spatiotemporal patterns of

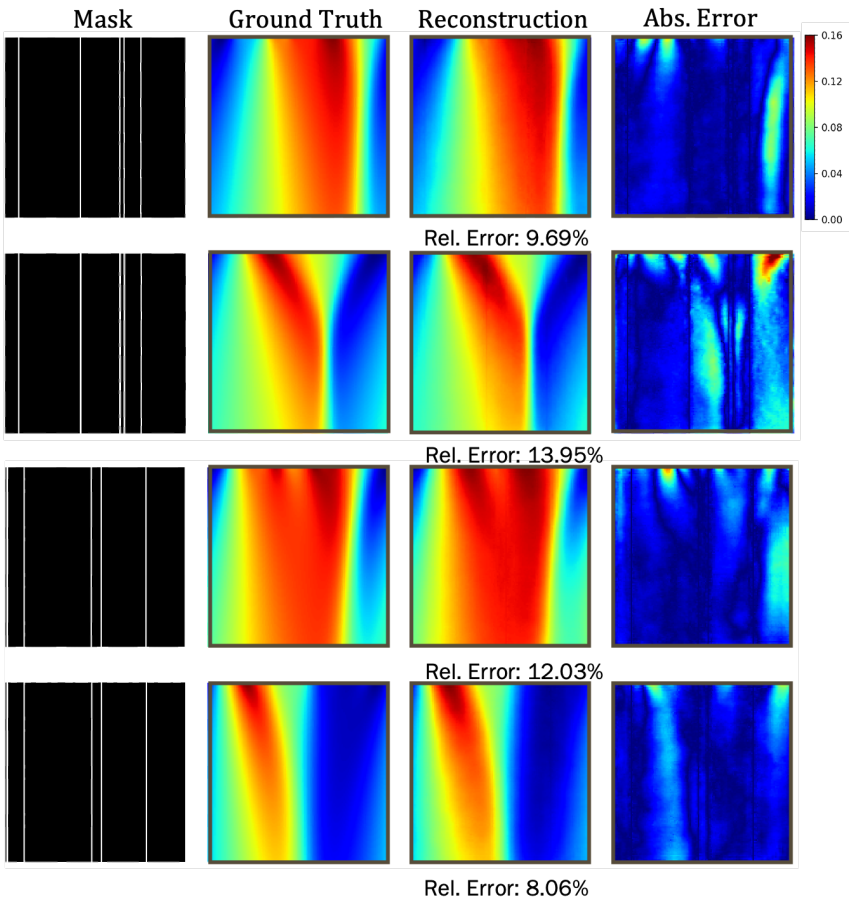


Figure 8: Qualitative result for the reconstruction problem of Burgers' equation. The mask column shows where the 5 sensors are placed (3.9% partial observation). The fourth column shows the absolute error $|\mathbf{x}^\star - \hat{\mathbf{x}}|$.

Table 17: Burgers' reconstruction problem. We report relative errors and PDE Error (↓). Step denotes the number of sampling steps per sample.

		Partial Observation	
Method	Step	Rel. Error	PDE Error ($\times 10^{-1}$)
Phys-Instruct	1	12.00%	7.59
DiffusionPDE	200	23.96%	9.24
CoCoGen	200	22.81%	8.00
PIDM	200	25.80%	7.91
DiffusionPDE	2000	4.66%	2.42
CoCoGen	2000	4.31%	1.78
PIDM	2000	6.59%	1.62

the ground truth, supporting the use of one-step sampling as a low-latency reconstruction baseline, which can serve as a strong initialization for subsequent refinement if desired.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction summarize the proposed Phys-Instruct framework, its scope, and the main empirical claims, which are supported by the theory and experiments.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

Justification: The paper discusses limitations and future directions, including the current fixed-resolution setting and extensions to resolution-robust or resolution-free models in Section 6.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes].

   Justification: The main theoretical result is stated in Theorem 4.1, with assumptions and the full proof provided in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes].

Justification: The paper provides benchmark definitions, training protocols, evaluation metrics, baselines, ablations, and implementation details in Section 5 and Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes].

Justification: We include the model code, dataset generation code, and reproduction instructions in the supplementary material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The paper reports the GPU types, training time, sampling latency, memory usage, and normalized compute breakdown in the main text and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: Stardard deviation results reported in Appendix F.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper reports the GPU types, training time, sampling latency, memory usage, and normalized compute breakdown in the main text and Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work uses PDE benchmarks and does not involve human subjects, sensitive personal data, or applications that conflict with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes].

    Justification: The work may positively impact scientific computing by enabling faster and more physically consistent generative solvers for forward, inverse, and reconstruction problems in PDE-based applications. Potential negative impacts are limited but include possible overreliance on high accuracy synthetic PDE data in high-stakes scientific or engineering settings, which we mitigate by emphasizing physics-consistency metrics and PDE-error evaluation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The paper does not release high-risk models or datasets such as language models, image generators for people, or scraped datasets.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A].

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A].

    Justification: The paper does not currently release new datasets, model checkpoints, or other standalone assets.

    Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The work does not involve crowdsourcing or human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: The work does not involve human subjects and therefore does not require IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: LLMs are not used as an important, original, or non-standard component of the core method or experiments.

    Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.